


中国研究生创新实践系列大赛

“华为杯”第十七届中国研究生数学建模竞赛

学　　校	湖南大学
参赛队号	20105320023
队员姓名	1.矫舒美
	2.姚天宇
	3.龙乙林

中国研究生创新实践系列大赛

“华为杯”第十七届中国研究生

数学建模竞赛

题 目 飞行器质心平衡供油策略优化研究

摘 要：

飞行器质心平衡供油策略优化是飞行器控制的一个重要课题。本文分析了初始油量、供油策略、俯仰角变化对飞行器的质心变化影响，建立了多油箱供油策略优化模型，利用带约束遗传算法求解出最优供油策略。

针对问题 1，首先建立**燃油质心定位模型**，燃油的质心定位可视为**多棱柱的中心定位**，因此根据 6 个油箱的供油速度与俯仰角变化，将燃油质心定位分为 4 类情况进行建模。然后根据物体质心计算公式建立**飞行器质心定位模型**，利用 Python 求得飞行器任意时刻的质心坐标，其瞬时质心位置变化曲线如图 4.7。

针对问题 2，以飞行器瞬时质心位置$\vec{c}_1(t)$与理想质心位置$\vec{c}_2(t)$的欧氏距离最大值达到最小为目标函数，考虑油箱供油的多个约束条件，建立**飞行器平飞-多油箱供油策略优化模型**，利用**带约束遗传算法**和**粒子群算法**分别得到 6 个油箱的供油策略。两种算法的**时间复杂度均为$O(n^3)$**，但带约束遗传算法的求解速度更快，所得结果更优。因此采用带约束遗传算法求得最优解，飞行器瞬时质心位置与理想质心位置的欧氏距离最大值为 **0.017566159*m***，4 个主油箱的总供油量为 **6463.992899*kg***。根据带约束遗传算法的求解结果，计算质心偏移量的平均值仅为 **0.011113916*m***，方差仅为 **1.31087×10^{-5}**，以上指标与粒子群算法的求解结果均验证了算法的**有效性**。

针对问题 3，基于问题 2 建立**飞行器平飞−多油箱供油分层优化模型**：第一层，优化 6 个油箱的初始油量；第二层，优化 6 个油箱的后续供油策略。首先利用**深度优先搜索算法**得到第一层最优结果，再代入第二层模型，利用**带约束遗传算法**得到分层优化结果。1～6 油箱初始载油量分别为**[0. 379413，1. 787226，2. 080872，1. 748531，2. 420809，0. 791332]*m*^{3}**、飞行器瞬时质心位置与理想质心位置的欧式距离最大值为 **0. 056718538*m***，4 个主油箱的总供油量 **6820.140521*kg***。分层优化算法的**时间复杂度为 *O(n^3)*。**根据求解结果，计算质心偏移量的平均值仅为 **0.025361366*m***，方差仅为 **5.22273×10^{-4}**，验证了算法的**有效性**。同时建立**全局优化模型**进行对比分析，验证了分层优化模型求解速度快，算法复杂度小，求解结果更优越。

针对问题 4，基于问题 2 考虑飞行器的俯仰角变化，采用问题 1 的飞行器质心定位模型，建立**飞行器俯仰-多油箱供油策略优化模型**，利用**带约束遗传算法**得到 6 个油箱的供油策略。飞行器瞬时质心位置与理想质心位置的欧氏距离最大值为 **0.069785212*m***，4 个主油

箱的总供油量为 **7035.54163kg**。该算法的**时间复杂度为$O(n^3)$**。本文根据求解结果，计算质心偏移量的平均值仅为 **0.020939828m**，方差仅为 **2.87969×10^{-4}**，验证了算法的**有效性**。

本文采用问题 2 的数据对**弃油量的上限值δ**做**灵敏度分析。**取δ的数量级为 10^{-3}、10^{-2}、10^{-1}、0，对比瞬时质心偏移量和最大质心偏移量得到：δ的取值会影响模型求解结果，且δ的数量级为 10^{-3} 时求解结果最优。

关键词：质心定位；带约束遗传算法；粒子群算法；分层优化；灵敏度分析

目录

1. 问题重述

1.1 问题背景

某一类飞行器携带有多个油箱，在飞行过程中，通过若干个油箱的联合供油以满足飞行任务要求和发动机工作需求。在任务执行过程中，飞行器的质心变化对飞行器的控制有着重要的影响，各个油箱内油量的分布和供油策略将导致飞行器质心的变化，进而影响到对飞行器的控制。因此，制定各油箱的供油策略是这类飞行器控制的一项重要任务，油箱的供油策略可用其向发动机或其它油箱供油的速度曲线来描述。

假设该类飞行器一共有 6 个油箱，各油箱供油示意图如图 1.1 所示：

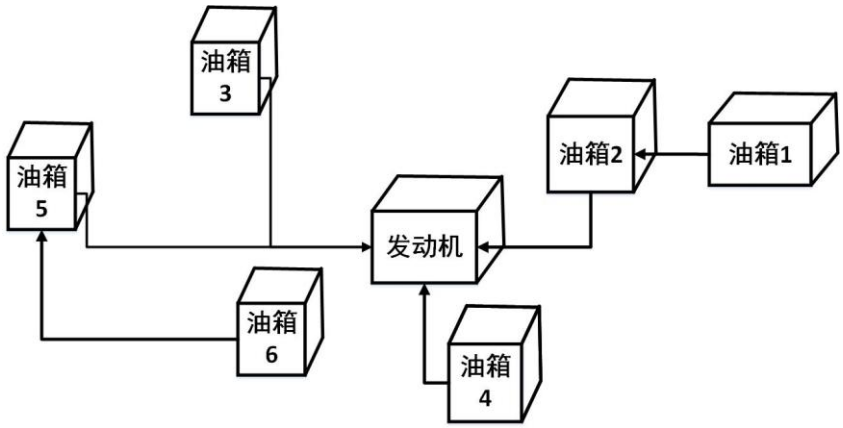


图 1.1 飞行器油箱供油示意图

1.2 需要解决的问题

问题 1：已知某次任务中飞行器的 6 个油箱的初始油量、供油速度及飞行器在飞行过程中的俯仰角变化数据，得出该飞行器在此次任务执行过程中，质心在飞行器坐标系下的位置变化数据，并画出质心变化曲线。

问题 2：附件 3 给出了某次任务的飞行器计划耗油速度数据，与飞行器在飞行器坐标系下的理想质心位置数据。根据任务需求，在飞行器始终保持平飞（俯仰角为 0）的任务规划过程中，为飞行器制定该次任务满足以下 6 个条件的 6 个油箱供油策略，使得飞行器每一时刻的质心位置$\vec{c}_1(t)$与理想质心位置$\vec{c}_2(t)$的欧氏距离的最大值达到最小。

条件（1）：油箱均为长方体且固定在飞行器内部，每个油箱内部长、宽、高的三个方向与飞行器坐标系的 x，y，z 轴三个方向平行。

条件（2）：在飞行器坐标系下，飞行器（不载油）质心为$\vec{c}_0$（0，0，0），第 i 个空油箱中心位置$\vec{P}_i$，i=1,2,⋯,6。飞行器（不载油）总重量为 M。

条件（3）：第 i 个油箱的供油速度上限为$U_i(U_i > 0)$，i=1,2,⋯,6。每个油箱一次供油的持续时间不少于 60 秒。

条件（4）：主油箱 2、3、4、5 可直接向发动机供油，油箱 1 和油箱 6 作为备份油箱分别为油箱 2 和油箱 5 供油，不能直接向发动机供油。

条件（5）：由于受到飞行器结构的限制，至多 2 个油箱可同时向发动机供油，至多 3 个油箱可同时供油。

条件（6）：飞行器在执行任务过程中，各油箱联合供油的总量应至少满足发动机的对耗油量的需要（若某时刻供油量大于计划耗油量，多余的燃油可通过其它装置排出飞行器）。

问题 3：假定初始油量未定，已知某次任务的飞行器计划耗油速度数据与飞行器在飞行器坐标系下的理想质心位置数据。在飞行器始终保持平飞（俯仰角为 0）的任务规划过程中，在满足问题 2 的 6 个条件的基础上，使得本次任务结束时 6 个油箱剩余燃油总量至少 1m^3，并且飞行器每一时刻的质心位置$\vec{c}_1(t)$与理想质心位置$\vec{c}_2(t)$的欧氏距离的最大值达到最小，确定飞行器该次任务的 6 个油箱初始载油量及供油策略。

问题 4：在实际任务规划过程中，飞行器俯仰角随时间变化。根据已有的飞行器俯仰角的变化数据和耗油速度数据，制定飞行器油箱供油策略，使得飞行器瞬时质心$\vec{c}_1(t)$与飞行器(不载油)质心$\vec{c}_0$的最大距离达到最小。

2. 模型假设

1、假设送到发动机的燃油会立即被消耗或者被排出飞行器，即飞行器（载油）的重量会相应减少。

2、忽略飞行器行驶过程中的惯性作用对燃油形状造成的影响。

3、假设油箱中油量小于某一限值ε且没有油量输入时，认为该油箱不再供油，则忽略条件（3）中对该油箱供油持续时间的约束。

3. 符号说明

符号	意义
i	油箱(i=1,2,3,4,5,6)、发动机(i=7)的序号
V_i^0	油箱 i 中燃油的初始体积
$V_i(t)$	t 时刻，油箱 i 中燃油的体积
$\theta(t)$	t 时刻，飞行器的俯仰角
$S_i(t)$	t 时刻，油箱 i 中燃油在 x-z 平面覆盖的面积
a_i	油箱 i 的长
b_i	油箱 i 的宽
c_i	油箱 i 的高
$a_i(t)$	t 时刻，油箱 i 中燃油在 x 轴上覆盖的长度
$c_i(t)$	t 时刻，油箱 i 中燃油在 z 轴上覆盖的长度
$x_i(t)$	t 时刻，油箱 i 中燃油质心在 x 轴方向上的坐标值
$y_i(t)$	t 时刻，油箱 i 中燃油质心在 y 轴方向上的坐标值
$z_i(t)$	t 时刻，油箱 i 中燃油质心在 z 轴方向上的坐标值
$\vec{c}_0$	飞行器（不载油）质心位置
$\vec{c}_1(t)$	t 时刻，飞行器实际质心位置
$\vec{c}_2(t)$	t 时刻，飞行器理想质心位置
$x_{c1}(t)$	t 时刻，飞行器的质心在 x 轴方向上的坐标值

$y_{c1}(t)$	t 时刻，飞行器的质心在 y 轴方向上的坐标值
$z_{c1}(t)$	t 时刻，飞行器的质心在 z 轴方向上的坐标值
m_i^0	油箱 i 中燃油的初始质量
$m_i(t)$	t 时刻，油箱 i 中燃油的质量
$\Delta m_i(t)$	t 时刻，油箱 i 供油量
ρ	燃油的密度
j	6 个油箱和 1 台发动机的序号 (j=7 为发动机)
$\vec{P}_i$	第 i 个空油箱中心位置
x_i^p	第 i 个空油箱中心位置在 x 轴方向上的坐标值
y_i^p	第 i 个空油箱中心位置在 y 轴方向上的坐标值
z_i^p	第 i 个空油箱中心位置在 z 轴方向上的坐标值
h_i^0	油箱 i 中燃油的初始高度
V_i	油箱 i 的体积
$\Delta V_i(t)$	t 时刻，油箱 i 中燃油体积的变化量
$v_i(t)$	t 时刻，油箱 i 的供油速度
$k_{ij}(t)$	t 时刻，油箱 i 向油箱/发动机 j 供油状态
τ_i	油箱 i 持续供油时长
x_i^0	油箱 i 中燃油初始质心在 x 轴方向上的坐标值
y_i^0	油箱 i 中燃油初始质心在 y 轴方向上的坐标值
z_i^0	油箱 i 中燃油初始质心在 z 轴方向上的坐标值
$\vec{d}_i^0$	油箱 i 的初始质心位置
$\vec{d}_i(t)$	油箱 i 中燃油在 t 时刻的质心位置
$v_h(t)$	t 时刻，飞行器的计划耗油速度
$v_{total}(t)$	t 时刻，4 个主油箱$(2, 3, 4, 5)$的总供油速度
m_{total}	4 个主油箱$(2, 3, 4, 5)$的总供油量
d_{max}	飞行器瞬时质心与理想质心距离的最大值
$D(t)$	t 时刻的质心偏移量（飞行器瞬时质心到理想质心（不载油质心）的欧氏距离）
Average	质心偏移量的平均值
Varp	质心偏移量的方差

4. 问题 1 的分析与求解

4.1 问题 1 分析

问题 1 要求根据 6 个油箱的供油速度以及飞行器的俯仰角变化计算每个时刻飞行器的质心坐标。根据物体质心计算公式可得飞行器的质心由飞行器（不载油）和各个燃油的质心决定。由于题目已给定飞行器（不载油）的质心，因此首先根据各个燃油的质量和俯仰角来计算各个油箱中燃油的质心坐标，最后根据各个燃油的质心坐标求得飞行器的瞬时质心位置。具体思路如下图 4.1 所示。

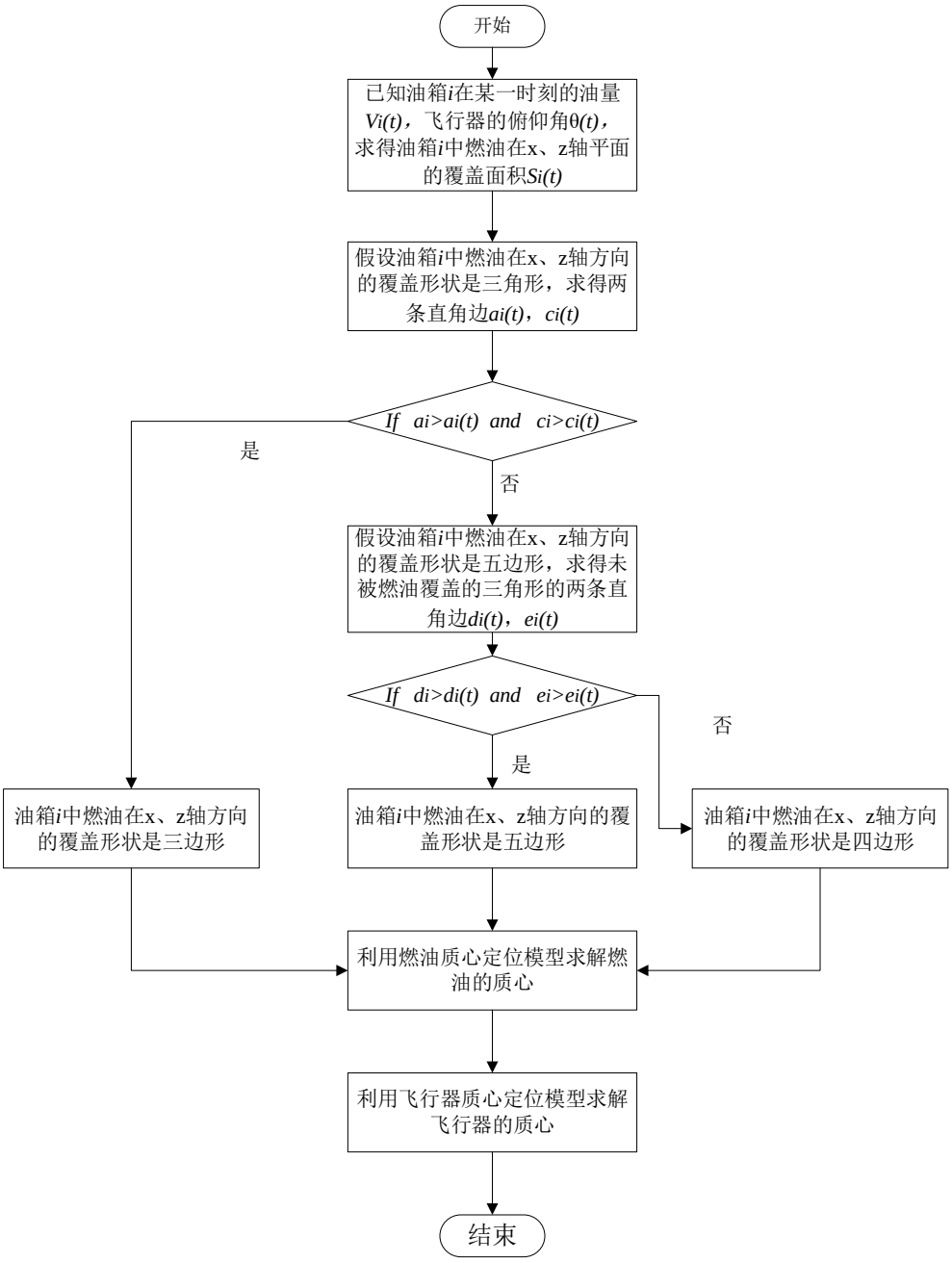


图 4.1　问题 1 思路导图

4.2 问题 1 建模

4.2.1 燃油质心定位模型

由于飞行器的姿态仅考虑俯仰变化，则燃油在油箱中的分布呈多棱柱形状，且 t 时刻油箱 i 中燃油质心在 y 轴方向上的坐标值$y_i(t)$保持不变，即$y_i(t)=y_i^P$，因此求燃油的质心转换为求棱柱底面多边形的重心，即在二维平面上求解$x_i(t)$和$z_i(t)$。

对于$x_i(t)$和$z_i(t)$，根据燃油的质量以及仰角，可以将燃油的质心计算分为 4 种情况，如图 4.2 所示。俯角的计算只是在仰角的基础上将x,z轴进行了互换，y轴旋转$180°$，因此本文只细述仰角情况。为了更好地区分 4 种情况，图 4.2 中采用 O-XYZ 坐标系（惯性坐标系），实际计算中采用飞行器坐标系 O(t)-X(t)Y(t)Z(t)。由几何关系易知计算$x_i(t)$和$z_i(t)$，首先计算底面多边形各个顶点的坐标，最后根据多边形的重心计算原理得到$x_i(t)$和$z_i(t)$。

多边形的重心计算原理[1]：将多边形分为 n 个三角形，第 i 个三角形的面积为σ_i，重心为(f_i,g_i),利用求平面薄板重心公式得到多边形的重心坐标(f,g)为

$$f=\frac{\sum_{i=1}^{n}f_i\sigma_i}{\sum_{i=1}^{n}\sigma_i},g=\frac{\sum_{i=1}^{n}g_i\sigma_i}{\sum_{i=1}^{n}\sigma_i} \tag{4.1}$$

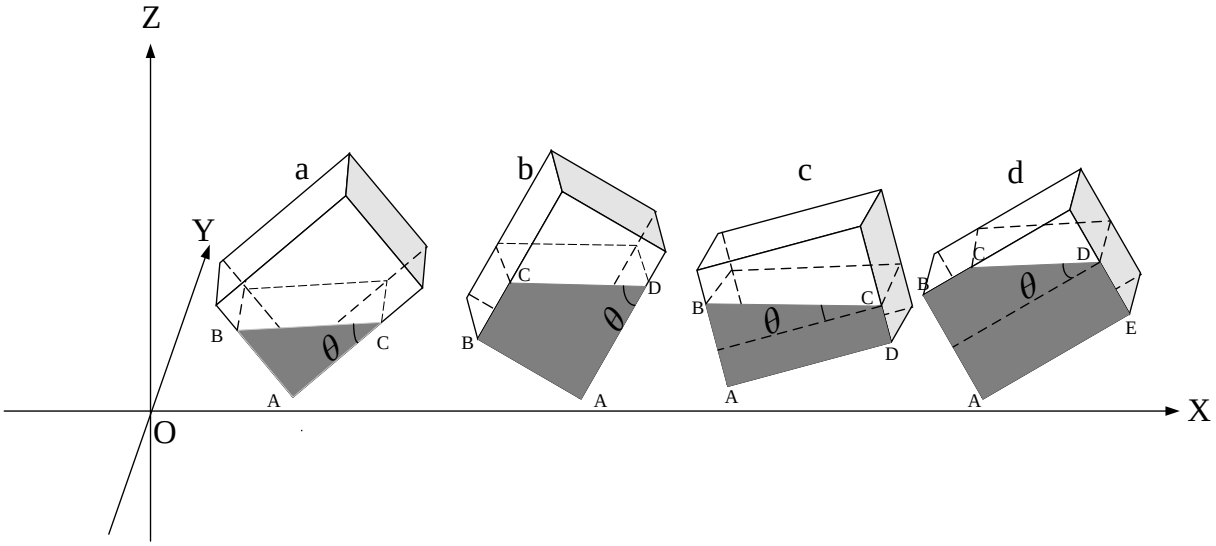


图 4.2　仰角的 4 种情况

（1）图 4.2 中 a 类情况的质心计算

$$V_i(t)=\frac{m_i(t)}{\rho} \tag{4.2}$$

$$S_i(t)=\frac{V_i(t)}{b_i} \tag{4.3}$$

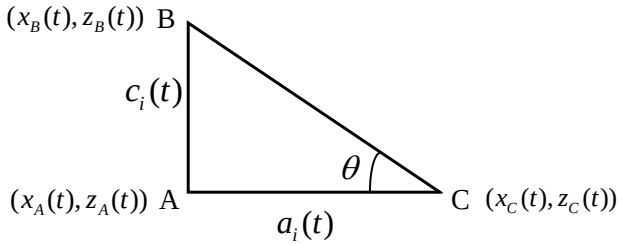


图 4.3　a 类情况底面多边形示意图

根据图 4.3 可得公式(4.4)和(4.5)，通过公式(4.2)-(4.5)即可求得$a_i(t)$和$c_i(t)$。A 点的坐标可根据附件 1 中给出的数据求得，从而通过 A 点的坐标、$a_i(t)$和$c_i(t)$即可求得 B、C 点的坐标。

$$\frac{c_i(t)}{a_i(t)} = \tan\theta(t) \tag{4.4}$$

$$\frac{c_i(t)\cdot a_i(t)}{2} = S_i(t) \tag{4.5}$$

因此对于 a 类情况，其燃油的质心坐标公式如下：

$$\begin{cases} x_i(t) = \dfrac{x_A(t) + x_B(t) + x_C(t)}{3} \\ y_i(t) = y_i^p \\ z_i(t) = \dfrac{z_A(t) + z_B(t) + z_C(t)}{3} \end{cases} \tag{4.6}$$

（2）图 4.2 中 b 类情况的质心计算

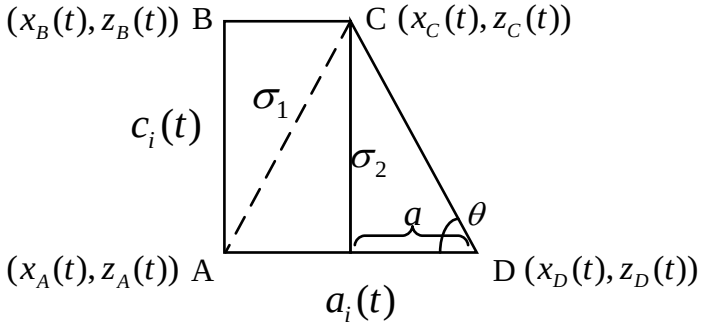


图 4.4　b 类情况底面多边形示意图

由图 4.2 和图 4.4 可得 b 类情况有以下几何关系

$$c_i(t) = c_i \tag{4.7}$$

$$a = \frac{c_i(t)}{\tan\theta(t)} \tag{4.8}$$

$$\frac{\left[\left(a_i(t) - a\right) + a_i(t)\right]\cdot c_i(t)}{2} = S_i(t) \tag{4.9}$$

由公式(4.2)-(4.3)、(4.7)-(4.9)即可求得$a_i(t)$，A 点的坐标可根据附件 1 中给出的数据求得，从而通过 A 点的坐标、$a_i(t)$和$c_i(t)$即可 B、C、D 点的坐标。

由多边形的重心计算原理可将图 4.4 按虚线划分为 2 个三角形，再根据 a 类情况的重心计算公式即可求得：

$$\begin{cases} x_i(t)=\dfrac{\sum_{j=1}^{n} x_j(t)\sigma_j(t)}{S_i(t)} \\ y_i(t)=y_i^p \\ z_i(t)=\dfrac{\sum_{j=1}^{n} z_j(t)\sigma_j(t)}{S_i(t)} \end{cases} \tag{4.10}$$

（3）图 4.2 中 c 类情况的质心计算

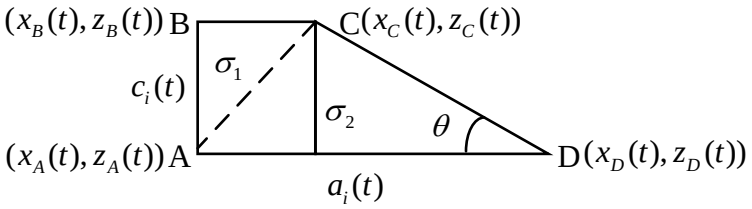


图 4.5　c 类情况底面多边形示意图

由图 4.2 和图 4.5 可得 c 类情况有以下几何关系

$$a_i(t)=a_i \tag{4.11}$$

$$\frac{\left[\left(a_i(t)-c_i(t)\cdot\tan\theta(t)\right)+a_i(t)\right]\cdot c_i(t)}{2}=S_i(t) \tag{4.12}$$

由公式(4.2)-(4.3)、(4.11)-(4.12)即可求得$c_i(t)$，A 点的坐标可根据附件 1 中给出的数据求得，从而通过 A 点的坐标、$a_i(t)$和$c_i(t)$即可 B、C、D 点的坐标。

由多边形的重心计算原理可将图 4.5 按虚线划分为 2 个三角形，再根据 a 类情况的重心计算公式和公式(4.10)即可求得 c 类情况的质心。

（4）图 4.2 中 d 类情况的质心计算

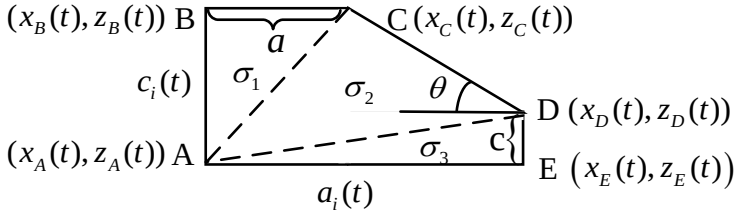


图 4.6　d 类情况底面多边形示意图

由图 4.2 和图 4.6 可得 d 类情况有以下几何关系

$$a_i(t)=a_i \tag{4.13}$$

$$c_i(t)=c_i \tag{4.14}$$

$$\frac{c_i(t)-c}{a_i(t)-a}=\tan\theta(t) \tag{4.15}$$

$$\left[\left(c_i(t)\cdot a_i(t)\right)-\frac{\left(c_i(t)-c\right)\cdot\left(a_i(t)-a\right)}{2}\right]=S_i(t) \tag{4.16}$$

由公式(4.2)-(4.3)、(4.13)-(4.16)即可求得a、c，A 点的坐标可根据附件 1 中给出的数据求得，从而通过 A 点的坐标、$a_i(t)$、$c_i(t)$、a、c即可 B、C、D、E 点的坐标。

由多边形的重心计算原理可将图 4.6 按虚线划分为 3 个三角形，再根据 a 类情况的重心计算公式和公式(4.10)即可求得 d 类情况的质心。

4.2.2 飞行器质心定位模型

为判断每个油箱在时刻 t 的供油状态，本文取决策变量$k_{ij}(t)$，当$k_{ij}(t)=1$，表示 t 时刻，油箱 i(i=1,2,…,6)向油箱/发动机 j(j=1,2,…,7)供油；当$k_{ij}(t)=0$，表示 t 时刻，油箱 i(i=1,2,…,6)没有向油箱/发动机 j(j=1,2,…,7)供油。

由于每秒会记录一组供油数据，所以油箱 i 在时刻t_s的供油速度$v_i(t_s)$相当于油箱 i 在时刻t_s的供油质量$m_i(t_s)$，且油箱 i 中燃油改变量$\Delta m_i(t_s)$：

$$\Delta m_i(t_s)=k_{ji}(t_s)v_j(t_s)-k_{iw}(t_s)v_i(t_s) \tag{4.17}$$

其中：$k_{ji}(t_s)$表示t_s时刻油箱 j 向油箱 i 的供油状态，$k_{iw}(t_s)$表示t_s时刻油箱 i 向油箱 w 的供油状态。在本问中，根据附件 2 已给出的供油速度数据，$k_{ji}(t_s)$、$k_{iw}(t_s)$均可以确定。

根据题目条件已知，$M=3000kg$，$\rho=850kg/m^3$，油箱 i 中燃油的质量$m_i(t)$计算公式如下：

$$m_i^0=V_i^0\cdot\rho \tag{4.18}$$

$$m_i(t)=m_i(t-1)+\Delta m_i(t)=m_i^0+\sum_{t_s=1}^{t}\Delta m_i(t_s),(t\geq 1) \tag{4.19}$$

根据物体质心定位原理可得，飞行器瞬时质心坐标由飞行器（不载油）的质心坐标、飞行器（不载油）的质量、各个油箱中燃油的质心坐标以及各个油箱中燃油的质量决定。4.2.1 已给出燃油的质心定位模型，且飞行器（不载油）的质心坐标为$\vec{c}_0$ =(0，0，0)，因此飞行器的质心位置$\vec{c}_1(t)$定位模型如下：

$$\begin{cases} x_{c1}(t)=\dfrac{\sum_{i=1}^{6}x_i(t)\cdot(m_i^0+\sum_{t_s=1}^{t}k_{ji}(t_s)v_j(t_s)-k_{iw}(t_s)v_i(t_s))}{M+\sum_{i=1}^{6}(m_i^0+\sum_{t_s=1}^{t}k_{ji}(t_s)v_j(t_s)-k_{iw}(t_s)v_i(t_s))} \\ y_{c1}(t)=\dfrac{\sum_{i=1}^{6}y_i(t)\cdot(m_i^0+\sum_{t_s=1}^{t}k_{ji}(t_s)v_j(t_s)-k_{iw}(t_s)v_i(t_s))}{M+\sum_{i=1}^{6}(m_i^0+\sum_{t_s=1}^{t}k_{ji}(t_s)v_j(t_s)-k_{iw}(t_s)v_i(t_s))} \\ z_{c1}(t)=\dfrac{\sum_{i=1}^{6}z_i(t)\cdot(m_i^0+\sum_{t_s=1}^{t}k_{ji}(t_s)v_j(t_s)-k_{iw}(t_s)v_i(t_s))}{M+\sum_{i=1}^{6}(m_i^0+\sum_{t_s=1}^{t}k_{ji}(t_s)v_j(t_s)-k_{iw}(t_s)v_i(t_s))} \end{cases} \tag{4.20}$$

4.3 模型求解

根据 4.2 的模型，利用 Python 可求得飞行器瞬时质心在飞行器坐标系下的位置数据，具体位置数据已写入表附件 6“第一问结果”中。飞行器的质心变化曲线如图 4.7 所示。

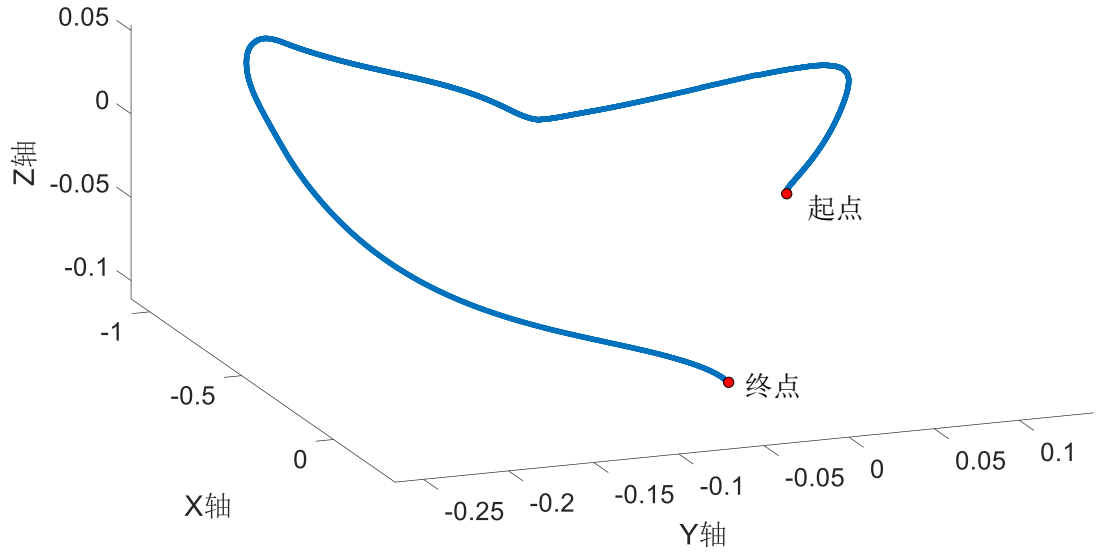


图 4.7　飞行器瞬时质心位置变化曲线

5. 问题 2 的分析与求解

5.1 问题 2 分析

问题 2 需要根据给出的飞行器计划耗油速度数据与飞行器在飞行器坐标系下的理想质心位置数据，求解出飞行器始终保持平飞状态下，满足相应约束条件的 6 个油箱供油策略，使得飞行器每一时刻的质心位置$\vec{c}_1(t)$与理想质心位置$\vec{c}_2(t)$的欧氏距离的最大值达到最小。首先采用问题 1 的飞行器质心定位模型求解飞行器瞬时质心坐标。在此基础上，本文建立飞行器平飞-多油箱供油策略优化模型，以飞行器每一时刻的质心位置$\vec{c}_1(t)$与理想质心位置$\vec{c}_2(t)$的欧氏距离的最大值达到最小为目标函数，满足相应 6 个约束条件，利用带约束的遗传算法[2,3]和粒子群算法分别得到 6 个油箱的供油策略。

5.2 飞行器平飞-多油箱供油策略优化模型

5.2.1 目标函数

为使得飞行器在每一时刻的质心位置$\vec{c}_1(t)$与理想质心位置$\vec{c}_2(t)$的欧氏距离的最大值达到最小，即目标函数为：

$$\min \max_t \left\| \vec{c}_1(\mathrm{t}) - \vec{c}_2(\mathrm{t}) \right\|_2 \tag{5.1}$$

飞行器理想质心位置$\vec{c}_2(t)$已在附件 3 给出，瞬时质心位置$\vec{c}_1(t)$求解方法如下。

已知每个油箱所装的初始燃油体积，则油箱 i 中燃油的初始高度为：

$$h_i^0 = \frac{V_i^0}{a_i \cdot b_i} \tag{5.2}$$

由于飞行器始终保持平飞，油箱 i 中燃油质心仅会随燃油的输入量和输出量之差的改变而在 z 轴方向上移动，但燃油质心在 x、y 轴保持不变，则可先求得油箱 i 的初始质心位置$\vec{d}_i^0(x_i^0, y_i^0, z_i^0)$，即

$$\begin{cases} x_i^0 = x_i^p \\ y_i^0 = y_i^p \\ z_i^0 = z_i^p + \dfrac{h_i^0}{2} - \dfrac{c_i}{2} \end{cases} \tag{5.3}$$

油箱 i 中燃油高度随燃油量的变化而在 z 轴方向上移动，则油箱 i 中燃油高度在t_s时刻的改变量为：

$$\Delta h_i(t_s) = \frac{\Delta V_i(t_s)}{a_i \cdot b_i} = \frac{\Delta m_i(t_s)}{\rho \cdot a_i \cdot b_i} = \frac{k_{ji}(t_s)v_j(t_s) - k_{iw}(t_s)v_i(t_s)}{\rho \cdot a_i \cdot b_i} \tag{5.4}$$

油箱 i 中燃油在 t 时刻的高度为：

$$h_i(t) = h_i^0 + \sum_{t_s=1}^{t} \Delta h_i(t_s) \tag{5.5}$$

则油箱 i 中燃油在 t 时刻的质心$\vec{d}_i(t)$为：

$$\vec{d}_i(\mathrm{t}) = (x_i^p, y_i^p, z_i^p + \frac{h_i^0 + \sum_{t_s=1}^{t} \Delta h_i(t_s)}{2} - \frac{c_i}{2}) \tag{5.6}$$

根据上式求得飞行器保持平飞时各油箱中燃油的质心位置，根据问题 1 的飞行器质心定位模型可求得飞行器平飞时的飞行器质心位置$\vec{c}_1(t) = (x_{c1}(t), y_{c1}(t), z_{c1}(t))$，计算公式如下：

$$\begin{cases} x_{c1}(t) = \dfrac{\sum_{i=1}^{6} x_i^p \cdot (m_i^0 + \sum_{t_s=1}^{t} k_{ji}(t_s)v_j(t_s) - k_{iw}(t_s)v_i(t_s))}{M + \sum_{i=1}^{6} (m_i^0 + \sum_{t_s=1}^{t} k_{ji}(t_s)v_j(t_s) - k_{iw}(t_s)v_i(t_s))} \\ y_{c1}(t) = \dfrac{\sum_{i=1}^{6} y_i^p \cdot (m_i^0 + \sum_{t_s=1}^{t} k_{ji}(t_s)v_j(t_s) - k_{iw}(t_s)v_i(t_s))}{M + \sum_{i=1}^{6} (m_i^0 + \sum_{t_s=1}^{t} k_{ji}(t_s)v_j(t_s) - k_{iw}(t_s)v_i(t_s))} \\ z_{c1}(t) = \dfrac{\sum_{i=1}^{6} [\left(z_i^p + \dfrac{h_i^0 + \sum_{t_s=1}^{t} \Delta h_i(t_s)}{2} - \dfrac{c_i}{2} \right) \cdot (m_i^0 + \sum_{t_s=1}^{t} k_{ji}(t_s)v_j(t_s) - k_{iw}(t_s)v_i(t_s))]}{M + \sum_{i=1}^{6} (m_i^0 + \sum_{t_s=1}^{t} k_{ji}(t_s)v_j(t_s) - k_{iw}(t_s)v_i(t_s))} \end{cases} \tag{5.7}$$

5.2.2 约束条件

（1）第 i 个油箱的供油速度上限为U_i（U_i>0），$i = 1,2,\cdots,6$，即

$$0 \le v_i(t) \le U_i \tag{5.8}$$

（2）每个油箱的油量约束：

$$m_i(t) \geq 0 \tag{5.9}$$

（3）为了确保每个油箱中的燃油一次供油时间不少于 60 秒，油箱 i 在出油时段里任意时刻的出油量要小于前一时刻的剩余燃油量；若油箱中油量小于某一限值ε且没有油量输入时，认为该油箱不再供油，则忽略条件(3)中对该油箱供油持续时间的约束，则有：

$$\begin{cases} m_i(t-1) > v_i(t) \\ \prod_{k=0}^{\tau_i} k_{ij}(t+k) = 1 \\ \begin{cases} \tau_i \geq 60, & m_i(t) > \varepsilon \\ \tau_i < 60, & m_i(t) < \varepsilon \end{cases} \end{cases} \tag{5.10}$$

（4）主油箱 2、3、4、5 可直接向发动机供油，油箱 1 和油箱 6 作为备份油箱分别为油箱 2 和油箱 5 供油，不能直接向发动机供油，即仅有如下表 5. 1 的供油关系实际存在，$k_{ij}(t)$可以为 1；而其余供油关系并不存在，其相对应的$k_{ij}(t)$均一直为 0。

表 5.1 油箱、发动机间的供油关系

i	j	$k_{ij}(t)$
1	2	0/1
2	7	0/1
3	7	0/1
4	7	0/1
5	7	0/1
6	5	0/1

由于油箱 2 和油箱 5 既有燃油的输出，又有燃油的输入，为保证油箱 2 和油箱 5 在任意 t 时刻可以装下油箱 1 和油箱 6 输送的燃油，建立如下约束：

$$V_i^0 + \sum_{t_s=1}^{t} \Delta V_i(t_s) \leq V_i \qquad i = 2,5 \tag{5.11}$$

（5）由于受到飞行器结构的限制，至多 2 个油箱可同时向发动机供油，至多 3 个油箱可同时供油，则有约束如下：

$$\begin{cases} \sum_{i=2}^{5} k_{i7}(t) \leq 2 \\ \sum_{i=1}^{6} \sum_{j=1}^{7} k_{ij}(t) \leq 3 \end{cases} \tag{5.12}$$

（6）飞行器进行飞行任务时，各油箱联合供油的总量应至少满足发动机的对耗油量的需要，同时为了保证各油箱可以在飞行器耗油阶段有油可供，本模型设置飞行器弃油上限δ，则有：

$$v_h \leq \sum_{i=2}^{5} k_{i7}(t) v_i(t) \leq v_h + \delta \tag{5.13}$$

5.3 基于带约束遗传算法的模型求解

5.3.1 带约束遗传算法

带约束遗传算法具体步骤如下：

步骤 1：读取飞行器的参数：油箱尺寸大小，油箱初始位置，油箱供油限制等。读取计划耗油曲线以及理想质心情况。

步骤 2：设置迭代数 ZI=10，设置遗传算法的参数：KI=100，种群数=1000，交叉、突变概率 0.8。设置 0-1 变量控制油箱供油时限。

步骤 3：在遗传算法产生初始种群时添加约束条件，产生满足变量约束的初始种群，计算各个体的适应度。

步骤 4：选出此代的较优个体作为亲代，进行交叉变异，产生子代，判断子代是否满足约束条件；若满足则 k=k+1，若不满足，则重新生成子代，直至子代数等于初始种群数。

步骤 5：判断遗传算法迭代是否结束？若未结束，则返回步骤 3，计算适应度，若结束，则根据遗传算法最优解，更新 0-1 状态变量，S=S+1，并存储上一时刻的最优解。

步骤 6：更新算法迭代数 k=0，油箱内油的质量，更新油耗速度、理想质心情况。若 S<=7200，则返回步骤三，进行下一时刻的优化，否则，z=z+1;

步骤 7：若 z<ZI,则返回步骤 2，初始化所有参数，进行计算，否则，输出计算结果，结束计算。算法流程图如图 5.1 所示。

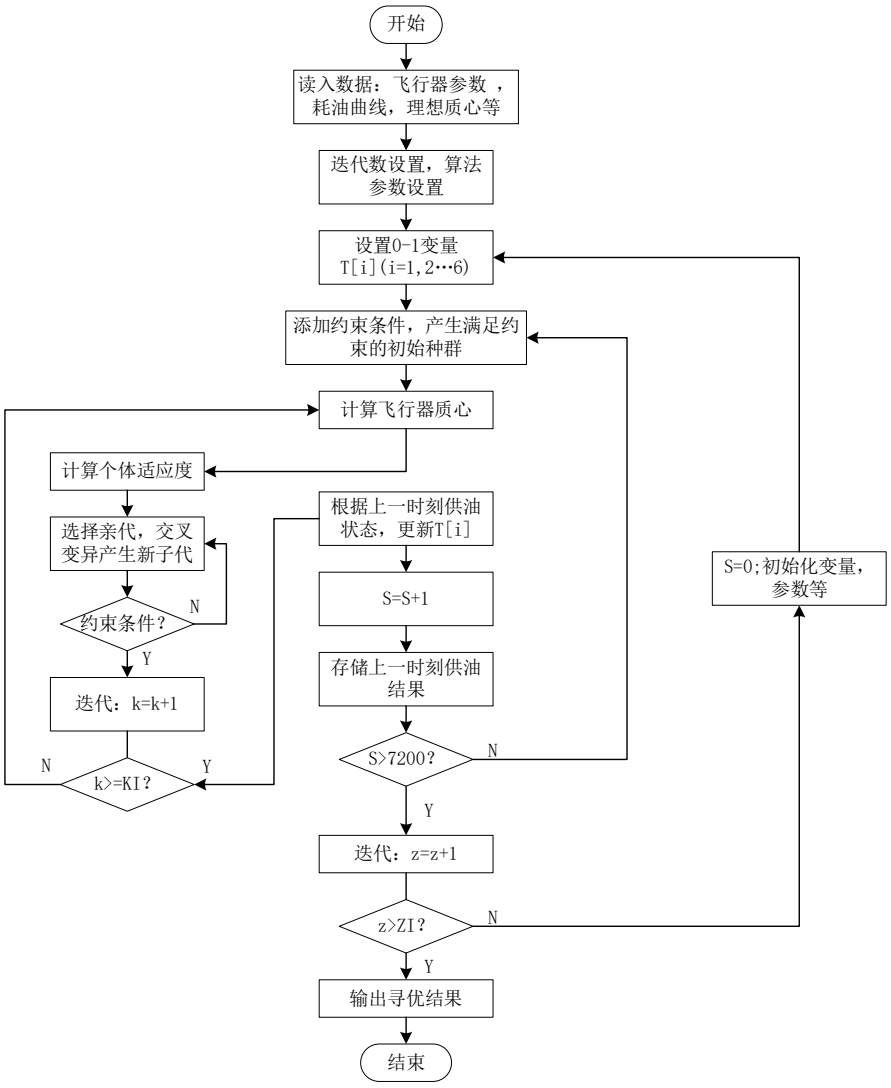


图 5.1　带约束遗传算法流程图

5.3.2 求解结果

基于飞行器平飞-多油箱供油策略优化模型，利用带约束遗传算法求解出 6 个油箱在任意时刻的供油速度，得到最优供油策略，结果如下。限值$\varepsilon=0.1$，$\delta=5\times10^{-3}$。本文带约束遗传算法参数设置如下：种群大小为 1000，终止进化代数为 200，交叉概率为 0.8，变异概率为 0.8。

（1）图 5.2 为 6 个油箱的供油速度曲线，图 5.3 为 4 个主油箱的总供油速度曲线。

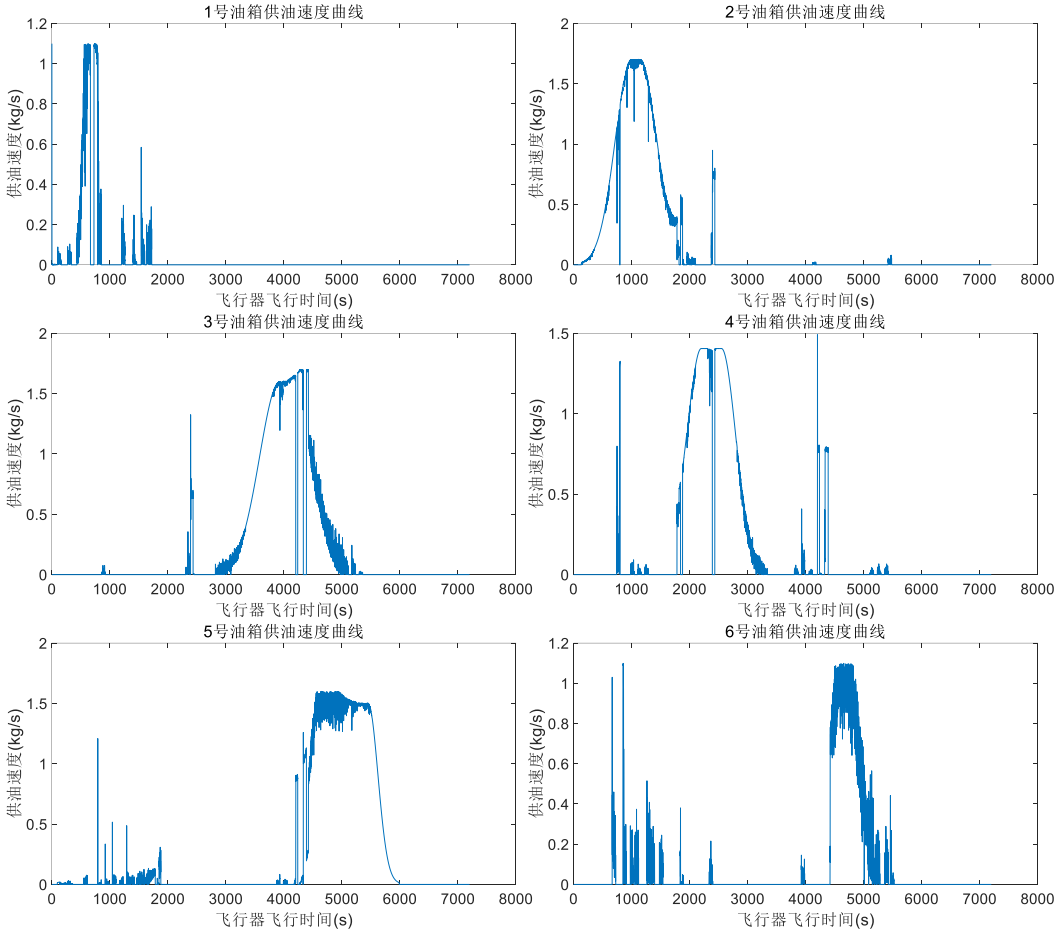


图 5.2　带约束遗传算法求解的 6 个油箱供油速度曲线

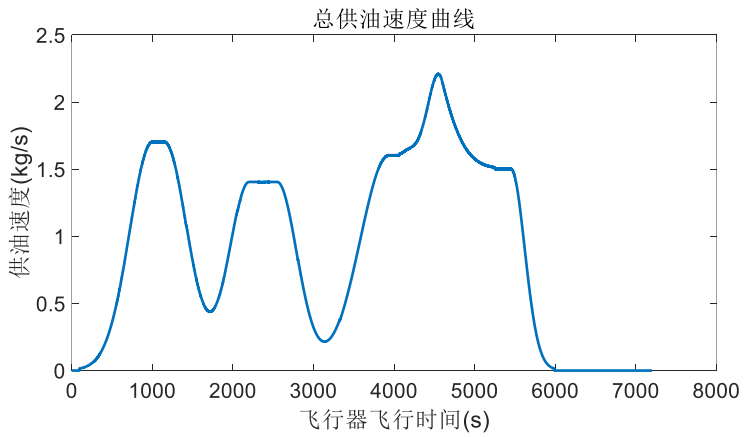


图 5.3　带约束遗传算法求解的 4 个主油箱总供油速度曲线

（2）每一时刻质心位置$\vec{c}_1(t)$与理想质心位置$\vec{c}_2(t)$欧氏距离的最大值为：

$$d_{\max} = 0.017566159(m) \tag{5.14}$$

（3）4 个主油箱的总供油量为：

$$m_{total} = 6463.992899(kg) \tag{5.15}$$

5.4 基于粒子群算法的模型求解

5.4.1 粒子群算法

粒子群算法（Particle Swarm Optimization，PSO）是一种基于群体的随机优化技术[4]，它将每个可能产生的解表述为群中的每一个微粒，每个微粒都具有自己的位置向量和速度向量，以及一个由目标函数决定的适应度，所有微粒在搜索空间中以一定速度飞行，在初始化一组随机解后通过迭代搜寻最优解，图 5. 4 是本文采用的粒子群算法流程。

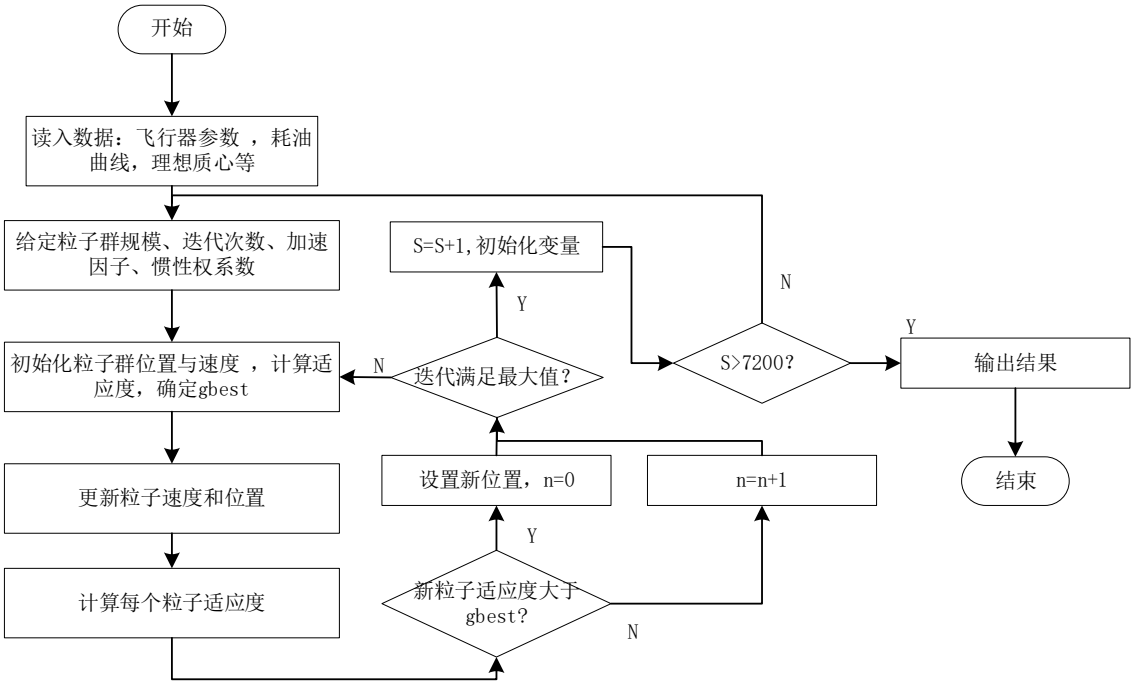


图 5.4　粒子群算法流程图

5.4.2 求解结果

基于飞行器平飞-多油箱供油策略优化模型，利用粒子群算法求解出 6 个油箱在任意时刻的供油速度，得到最优供油策略，结果如下。限值$\varepsilon = 0.1$，$\delta = 5 \times 10^{-3}$。本文粒子群算法参数设置如下：种群大小为 3000，迭代次数为 200，惯性权值为 0.9，加速因子$c_1 = c_2 = 0.8$。

（1）图 5.5 为 6 个油箱的供油速度曲线，图 5.6 为 4 个主油箱的总供油速度曲线。

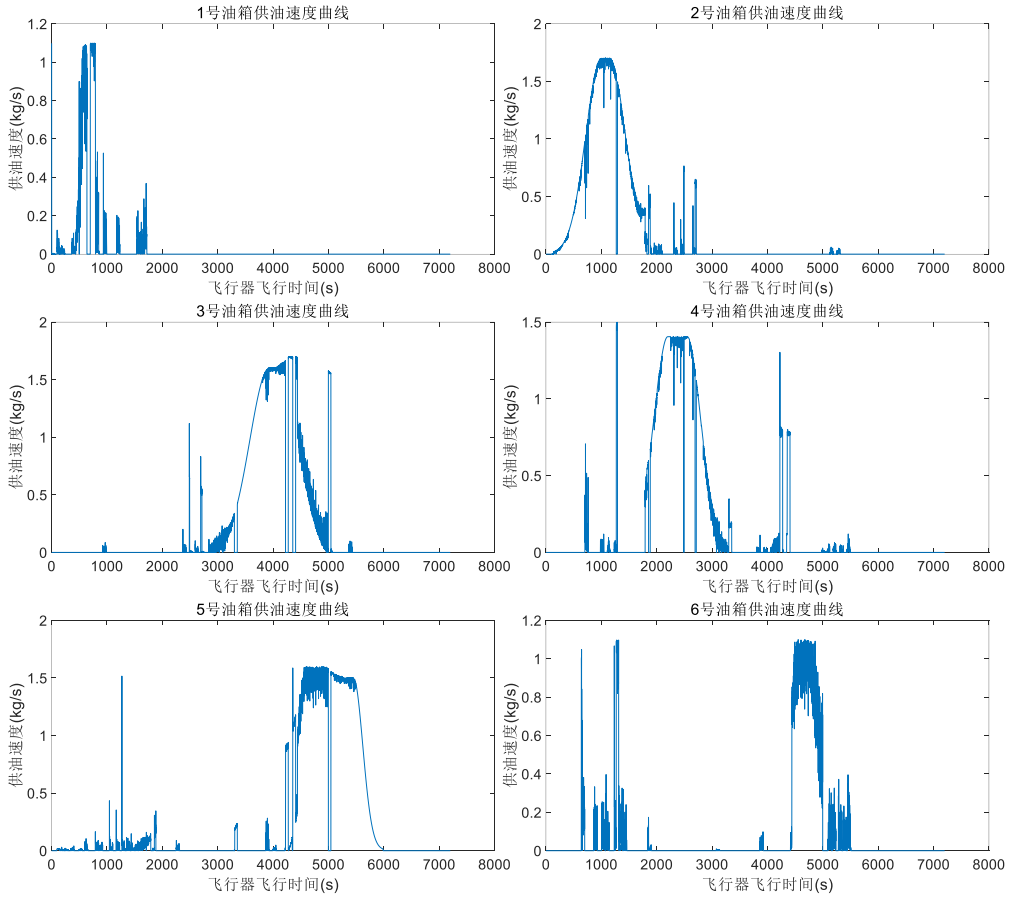


图 5.5　粒子群算法求解的 6 个油箱供油速度曲线

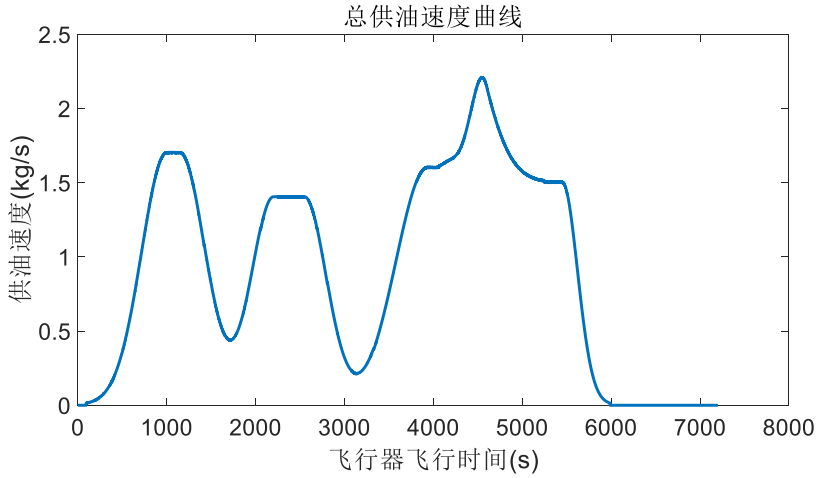


图 5.6　粒子群算法求解的 4 个主油箱总供油速度曲线

（2）每一时刻的质心位置$\vec{c}_1(t)$与理想质心位置$\vec{c}_2(t)$的欧氏距离的最大值为：

$$d_{\max} = 0.056036349(m) \tag{5.16}$$

（3）4 个主油箱的总供油量为：

$$m_{total} = 6463.540992(kg) \tag{5.17}$$

对比两种算法的最优解d_{max}，可知带约束遗传算法的求解结果更优越，因此本文选用带约束遗传算法求解的 6 个油箱供油速度数据写入附件 6“第二问结果”中。

5.5 算法有效性和复杂度

本文根据求解结果，得到飞行器任意时刻的质心偏移量$D(t)$。计算质心偏移量的平均值和方差验证算法的有效性。

$$D(t)=\left\|\overrightarrow{c_1}(t)-\overrightarrow{c_2}(t)\right\|_2 \tag{5.18}$$

$$Average=\frac{\sum_{t=1}^{7200}D(t)}{7200} \tag{5.19}$$

$$Varp=\sqrt{\frac{\sum_{t=1}^{7200}(D(t)-Average)^2}{7200}} \tag{5.20}$$

其中，*Avarege* 为质心偏移量的平均值，*Varp* 为质心偏移量的方差。

由带约束遗传算法与粒子群算法分别求得质心偏移量的平均值$Average$、方差值$Varp$、运行时长以及对应的算法复杂度如下表 5. 2 所示。

表 5. 2　两种算法对比

算法类别	$Average/m$	$Varp$	运行时长（min）	算法复杂度
带约束遗传算法	0.011113916	1.31087×10^{-5}	≈4	$O(n^3)$
粒子群算法	0.023065116	3.96362×10^{-4}	≈300	$O(n^3)$

分析以上数据，可以得出以下结论：

（1）两种算法得到质心偏移量的平均值与方差都很小，粒子群算法得出的 6 个油箱供油策略与带约束遗传算法的结果也相差不大，这说明两种算法均可以得到可靠并有效的结果，且证明了本文带约束遗传算法的求解有效性，提高了结果可信度。

（2）两种算法的复杂度一致，但带约束遗传算法的运行时长远小于粒子群算法。

（3）带约束遗传算法的解优于粒子群算法，且其质心偏移量方差值比粒子群算法小得多，说明带约束遗传算法更适合求解本文模型。因此本文后续模型均采用带约束遗传算法求解。

6. 问题 3 的分析与求解

6.1 问题 3 分析

问题 3 相较于问题 2，初始油量未给定，且要求任务结束时 6 个油箱中的总剩余燃油量不少于 1m^3。根据新增条件，问题 3 即在问题 2 的飞行器平飞−多油箱供油策略优化模型中新增 6 个变量，1 个约束条件。由于初始油量会影响后续的供油策略，为减小计算复杂度，本文对问题 3 的模型进行分层优化[5], 首先考虑 6 个油箱的初始油量，再确定 6 个油箱的供油策略。但为验证分层优化模型在求解时效率更高，更有灵活性，本文建立全局优化模型进行对比验证。

本节分层优化模型的优化策略分为两层：

第一层：优化 6 个油箱的初始油量，使得$\|\vec{c}_1(1)-\vec{c}_2(1)\|_2$最小。

第二层：优化 6 个油箱的后续供油策略，使得$\max_t\|\vec{c}_1(t)-\vec{c}_2(t)\|_2$最小。

6.2 飞行器平飞-多油箱供油分层优化模型

6.2.1 第一层模型

第一层模型的优化目标为$\|\vec{c}_1(1)-\vec{c}_2(1)\|_2$最小，同时需满足各个油箱的初始油量小于等于油箱体积，且问题 3 要求任务结束时 6 个油箱中的总剩余燃油量不少于 1m^3，因此 6 个油箱的总初始油量至少比总耗油量多 1m^3。

求 6 个油箱的初始油量，使：

$$obj.\ \min\left\|\vec{c}_1(1)-\vec{c}_2(1)\right\|_2 \tag{6.1}$$

$$s.t.\begin{cases} V_i^0 \le V_i \quad ,i=1,2,...,6 \\ \sum_{i=1}^{6} V_i^0 \ge \dfrac{\sum_{t=1}^{7200} v_h(t)}{\rho}+1 \end{cases} \tag{6.2}$$

6.2.2 第二层模型

确定了 6 个油箱的初始油量后，第二层数学模型与问题 2 数学模型基本一致。由于飞行器完成平飞任务时 6 个油箱中的总剩余燃油量不少于 1m^3，因此新增约束条件：

$$\sum_{i=1}^{6} V_i(7200) \ge 1 \tag{6.3}$$

求最优供油策略，使：

$$obj.\ \min \max_t \left\|\vec{c}_1(t)-\vec{c}_2(t)\right\|_2 \tag{6.4}$$

$$s.t.\begin{cases} \vec{c_1}(t)=\left(x_{c1}(t),y_{c1}(t),z_{c1}(t)\right) \\ x_{c1}(t)=\dfrac{\sum_{i=1}^{6} x_i^p\cdot(m_i^0+\sum_{t_s=1}^{t}k_{ji}(t_s)v_j(t_s)-k_{iw}(t_s)v_i(t_s))}{M+\sum_{i=1}^{6}(m_i^0+\sum_{t_s=1}^{t}k_{ji}(t_s)v_j(t_s)-k_{iw}(t_s)v_i(t_s))} \\ y_{c1}(t)=\dfrac{\sum_{i=1}^{6} y_i^p\cdot(m_i^0+\sum_{t_s=1}^{t}k_{ji}(t_s)v_j(t_s)-k_{iw}(t_s)v_i(t_s))}{M+\sum_{i=1}^{6}(m_i^0+\sum_{t_s=1}^{t}k_{ji}(t_s)v_j(t_s)-k_{iw}(t_s)v_i(t_s))} \\ z_{c1}(t)=\dfrac{\sum_{i=1}^{6}[\left(z_i^p+\frac{h_i^0+\sum_{t_s=1}^{t}\Delta h_i(t_s)}{2}-\frac{c_i}{2}\right)\cdot(m_i^0+\sum_{t_s=1}^{t}k_{ji}(t_s)v_j(t_s)-k_{iw}(t_s)v_i(t_s))]}{M+\sum_{i=1}^{6}(m_i^0+\sum_{t_s=1}^{t}k_{ji}(t_s)v_j(t_s)-k_{iw}(t_s)v_i(t_s))} \\ h_i^0=\dfrac{V_i^0}{x_{li}\cdot y_{li}} \\ m_i^0=V_i^0\cdot\rho \end{cases} \tag{6.5a}$$

$$
s.t.\begin{cases}
\Delta h_i(t_s)=\dfrac{\Delta V_i(t_s)}{x_{li}\cdot y_{li}}=\dfrac{\Delta m_i(t_s)}{\rho\cdot x_{li}\cdot y_{li}}=\dfrac{k_{ji}(t_s)v_j(t_s)-k_{iw}(t_s)v_i(t_s)}{\rho\cdot x_{li}\cdot y_{li}} \\
0\le v_i(t)\le U_i \\
m_i(t)\ge 0 \\
\begin{cases}
m_i(t-1)>v_i(t) \\
\prod_{k=0}^{\tau_i}k_{ij}(t+k)=1 \\
\begin{cases}
\tau_i\ge 60, & m_i(t)>\varepsilon \\
\tau_i<60, & m_i(t)<\varepsilon
\end{cases}
\end{cases} \\
V_i^0+\sum_{t_s=1}^{t}\Delta V_i(t_s)\le V_i \qquad ,i=2,5 \\
\begin{cases}
\sum_{i=2}^{5}k_{i7}(t)\le 2 \\
\sum_{i=1}^{6}\sum_{j=1}^{7}k_{ij}(t)\le 3
\end{cases} \\
v_h\le\sum_{i=2}^{5}k_{i7}(t)v_i(t)\le v_h+\delta \\
\sum_{i=1}^{6}V_i(7200)\ge 1
\end{cases}
\tag{6.5b}
$$

6.3 算法流程

由第 5 章的算法对比可知，带约束遗传算法在求解本文模型时更具有优越性，且问题 3、4 是在问题 2 基础上建立的数学模型，所以问题 3、4 均用带约束遗传算法进行求解，这里不再详细阐述带约束遗传算法，以下是问题 3 中两个数学模型的算法流程。

6.3.1 分层优化算法流程

带约束遗传算法分层优化求解流程如下：

步骤 1：读取飞行器的参数：油箱尺寸大小，油箱初始位置，油箱供油限制等。读取计划耗油曲线以及理想质心情况。

步骤 2：设置双层迭代的数 Z=10。

步骤 3：上层使用深度优先搜索算法，得到第一层的最优解。

步骤 4：将上层求解结果带入下层，使用问题 2 求解时使用的算法进行求解。

步骤 5：迭代数 z=z+1，将下层求解结果，若迭代数达到设置数目，则返回最优值并记录结果。

6.3.2 全局优化算法流程

不使用分层优化，直接全局优化目标函数，即目标函数仍为公式(6.1)，约束条件包括公式(6.2)-(6.5)，建立飞行器平飞−多油箱供油全局优化模型，此处不再赘述。

带约束遗传算法全局优化求解流程如下：

步骤 1：读取飞行器的参数：油箱尺寸大小，油箱初始位置，油箱供油限制等。读取计划耗油曲线以及理想质心情况。

步骤 2：利用遗传算法得到油箱初始油量，将初始油量带入问题 2 的模型中求解，得到相应的适应度。

步骤 3：找到最大适应度作为最佳求解结果，输出最优值。

6.4 两个模型对比

6.4.1 求解结果

基于上文建立的分层优化模型和全局优化模型，利用带约束遗传算法求解出 6 个油箱在任意时刻的供油速度，得到最优供油策略，结果如下。模型中，限值ε=0. 1，弃油量上限值δ=5×10^{-3}。

6 个油箱的初始载油量、飞行器瞬时质心与理想质心距离的最大值d_{max}、4 个主油箱的总供油量m_{total}，如表 6. 1-6. 2 所示。

表 6.1　6 个油箱的初始载油量

油箱编号	模型类别	1	2	3	4	5	6
载油量（m^3）	分层优化模型	0.379413	1.787226	2.080872	1.748531	2.420809	0.791332
	全局优化模型	0.301233	1.926163	2.106764	1.795364	2.546207	0.610163

表 6. 2　6 个油箱的d_{max}、m_{total}

模型类别	$d_{\max}$ / m	m_{total} / kg
分层优化模型	0. 056718538	6820. 140521
全局优化模型	0. 085198313	6821. 337356

利用分层优化模型求得飞行器瞬时质心与理想质心距离最大值d_{max}较全局优化所寻得的结果小 33. 43%，说明分层优化模型提供的油箱供油策略可使飞行器更加平稳的运行，且由附件 4 的数据可知飞行器的计划耗油量为 6805. 174669kg，可知由分层优化模型所求得的飞行器在空中弃油量 14. 945852kg，而全局最优模型求解到飞行器在空中弃油量为 16. 162687kg。综上可以看出：分层优化模型所得结果更加合理有效，因此本文选取分层优化模型作为问题 3 的求解模型，全局优化模型所得的 6 个油箱供油速度曲线，4 个主油箱的总供油速度曲线已附到附录，在此不做详述。

图 6. 1 为分层优化模型所得的 6 个油箱供油速度曲线，图 6. 2 为分层优化模型所得的 4 个主油箱的总供油速度曲线。分层优化模型所得的 6 个油箱初始载油量和供油速度数据已存入附件 6 结果表“第三问结果”中。

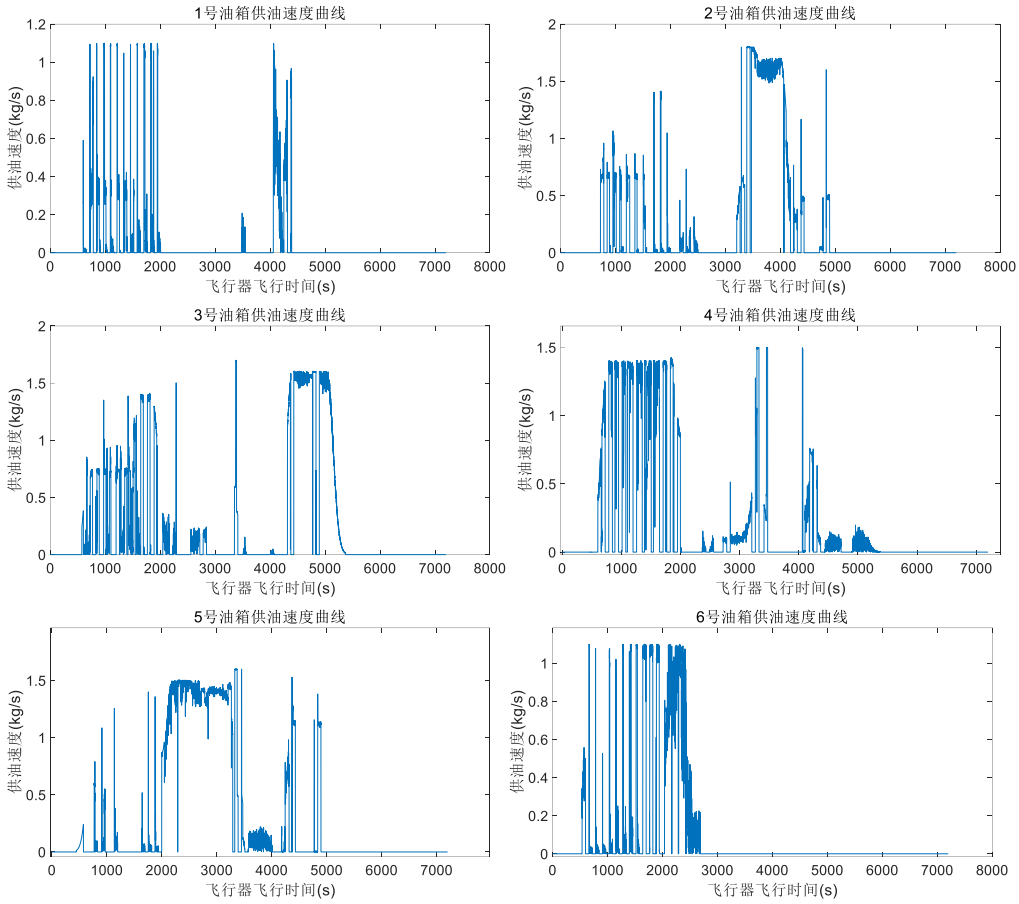


图 6.1　分层优化模型求解的 6 个油箱的供油速度曲线

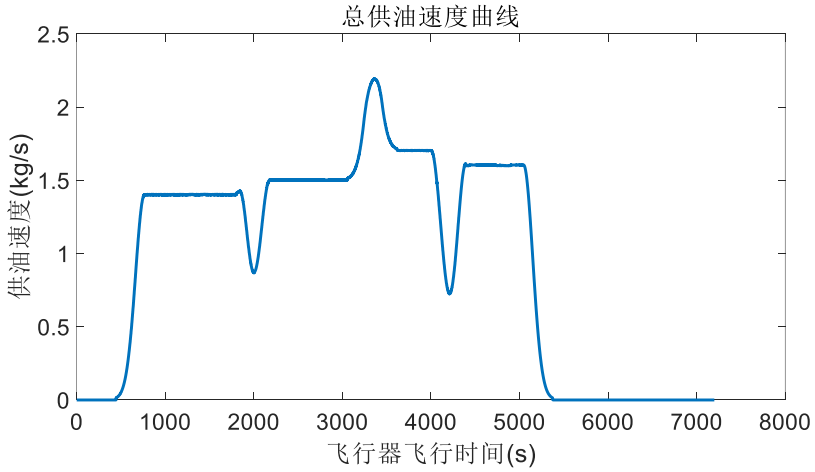


图 6.2　分层优化模型求解的 4 个主油箱总供油速度曲线

6.4.2 算法有效性和复杂度

由分层优化与全局优化分别求得飞行器质心偏移量的平均值$Average$、方差值$Varp$以及对应的算法复杂度如下表 6. 3 所示。

表 6.3　两个模型对比

模型类别	$Average/m$	$Varp$	算法复杂度
分层优化模型	0.025361366	5.22273×10^{-4}	$O(n^3)$
全局优化模型	0.038110445	8.28389×10^{-4}	$O(n^4)$

分析以上数据，可以得出以下结论：

（1）两种模型得到的距离平均值相近，相差 $0.012749079m$，且分层优化模型得出的 6 个油箱供油策略与全局规划的结果也相差不大，这说明两个优化模型均可以得到可靠并有效的结果，证明了本文分层优化模型的科学性，提高了结果可信度。

（2）分层优化模型求解效率远高于全局优化模型，分层优化算法复杂度$O(n^3)$比全局优化算法复杂度$O(n^4)$低。

（3）分层优化模型的解优于全局优化模型，分层优化距离的结果方差值比全局优化的结果方差值小得多，说明分层优化的供油策略可以使飞行器飞行的更平稳。

7. 问题 4 的分析与求解

7.1 问题 4 分析

问题 4 要求根据俯仰角的变化和耗油速度，制定油箱供油策略，使得飞行器瞬时质心$\vec{c}_1(t)$与飞行器（不载油）质心$\vec{c}_0$的最大距离达到最小。本文采用问题 1 的飞行器质心定位模型，基于问题 2 建立飞行器俯仰−多油箱供油优化模型，以飞行器每一时刻的质心位置$\vec{c}_1(t)$与飞行器（不载油）质心$\vec{c}_0$的欧氏距离的最大值达到最小为目标函数，满足相应 6 个约束条件，利用带约束遗传算法得到 6 个油箱的供油策略。

7.2 飞行器俯仰-多油箱供油策略优化模型

根据问题 1 的飞行器质心定位模型可求得飞行器俯仰飞行的瞬时质心位置$\vec{c}_1(t)=(x_{c1}(t),y_{c1}(t),z_{c1}(t))$，计算公式如(4.20)所示。利用 Python 求解飞行器质心位置时，可直接调用求解问题 1 的函数。

为使得飞行器在每一时刻的质心位置$\vec{c}_1(t)$与飞行器（不载油）质心$\vec{c}_0$的欧氏距离的最大值达到最小，即目标函数为：

$$\min\max_t \left\|\vec{c}_1(t)-\vec{c}_0\right\|_2 \tag{7.1}$$

根据题目要求，问题 4 的约束条件与问题 2 的约束条件一致，因此问题 4 的数学模型如下：

$$obj.\min\max_t \left\|\vec{c}_1(t)-\vec{c}_0\right\|_2 \tag{7.2}$$

$$
s.t.\begin{cases}
\vec{c_1}(t)=\left(x_{c1}(t),y_{c1}(t),z_{c1}(t)\right)\\
x_{c1}(t)=\dfrac{\sum_{i=1}^{6}x_i(t)\cdot(m_i^0+\sum_{t_s=1}^{t}k_{ji}(t_s)v_j(t_s)-k_{iw}(t_s)v_i(t_s))}{M+\sum_{i=1}^{6}(m_i^0+\sum_{t_s=1}^{t}k_{ji}(t_s)v_j(t_s)-k_{iw}(t_s)v_i(t_s))}\\
y_{c1}(t)=\dfrac{\sum_{i=1}^{6}y_i(t)\cdot(m_i^0+\sum_{t_s=1}^{t}k_{ji}(t_s)v_j(t_s)-k_{iw}(t_s)v_i(t_s))}{M+\sum_{i=1}^{6}(m_i^0+\sum_{t_s=1}^{t}k_{ji}(t_s)v_j(t_s)-k_{iw}(t_s)v_i(t_s))}\\
z_{c1}(t)=\dfrac{\sum_{i=1}^{6}z_i(t)\cdot(m_i^0+\sum_{t_s=1}^{t}k_{ji}(t_s)v_j(t_s)-k_{iw}(t_s)v_i(t_s))}{M+\sum_{i=1}^{6}(m_i^0+\sum_{t_s=1}^{t}k_{ji}(t_s)v_j(t_s)-k_{iw}(t_s)v_i(t_s))}\\
h_i^0=\dfrac{V_i^0}{x_{li}\cdot y_{li}}\\
m_i^0=V_i^0\cdot\rho\\
\Delta h_i(t_s)=\dfrac{\Delta V_i(t_s)}{x_{li}\cdot y_{li}}=\dfrac{\Delta m_i(t_s)}{\rho\cdot x_{li}\cdot y_{li}}=\dfrac{k_{ji}(t_s)v_j(t_s)-k_{iw}(t_s)v_i(t_s)}{\rho\cdot x_{li}\cdot y_{li}}\\
0\le v_i(t)\le U_i\\
m_i(t)\ge 0\\
\begin{cases}
m_i(t-1)>v_i(t)\\
\prod_{k=0}^{\tau_i}k_{ij}(t+k)=1\\
\begin{cases}\tau_i\ge 60, & m_i(t)>\varepsilon\\ \tau_i<60, & m_i(t)<\varepsilon\end{cases}
\end{cases}\\
\begin{cases}
V_i^0+\sum_{t_s=1}^{t}\Delta V_i(t_s)\le V_i & ,\ i=2,5\\
V_i^0\le V_i & ,i=1,2,...,6
\end{cases}\\
\begin{cases}
\sum_{i=2}^{5}k_{i7}(t)\le 2\\
\sum_{i=1}^{6}\sum_{j=1}^{7}k_{ij}(t)\le 3
\end{cases}\\
v_h\le\sum_{i=2}^{5}k_{i7}(t)v_i(t)\le v_h+\delta
\end{cases}
\tag{7.3}
$$

7.3 模型求解及算法复杂度

由于问题 4 的算法流程与问题 2 基本一致，因此此处不做详述。基于飞行器俯仰-多油箱供油策略优化模型，利用带约束遗传算法求解出 6 个油箱在任意时刻的供油速度，得到最优供油策略，结果如下。限值$\varepsilon=0.1$，弃油量上限$\delta=5\times10^{-3}$。

（1）图 7.1 为 6 个油箱的供油速度曲线，图 7.2 为 4 个主油箱的总供油速度曲线和计划耗油速度曲线。

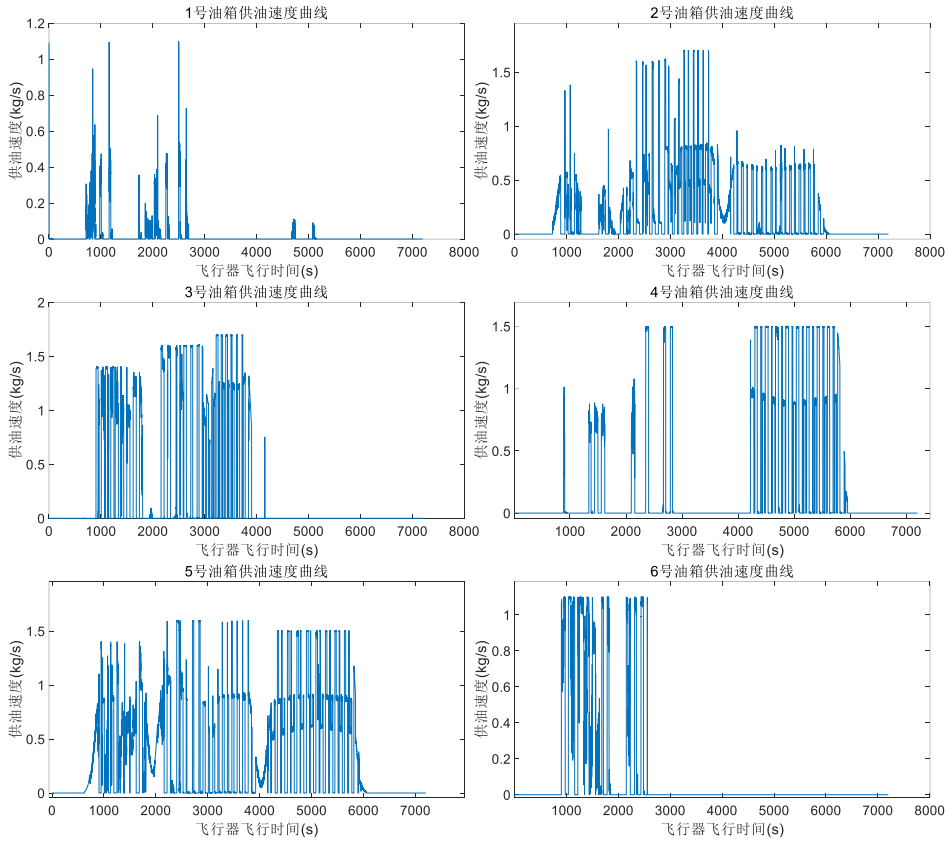


图 7.1　问题 4 的 6 个油箱供油速度曲线

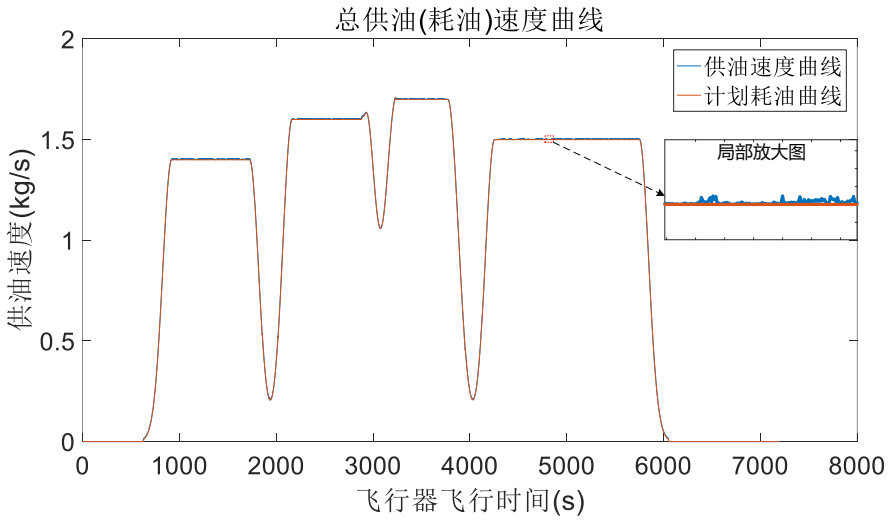


图 7.2　问题 4 的 4 个主油箱总供油速度曲线和计划耗油速度曲线

从图 7.2 可以得到每一时刻的供油量与耗油量之差很小，说明采用本文的数学模型与求解算法能减少弃油量，降低燃油成本。

（2）每一时刻的质心位置$\vec{c}_1(t)$与理想质心位置$\vec{c}_2(t)$的欧氏距离的最大值为：

$$d_{\max} = 0.069785212(m) \tag{7.4}$$

（3）4 个主油箱的总供油量为：

$$m_{total}=7035.545163(kg) \tag{7.5}$$

（4）6 个油箱供油速度数据已存入附件 6 结果表“第四问结果”中。

算法有效性和复杂度：根据结果可求得，$Avarege = 0.020939828\,m$，$Varp = 2.87969 \times 10^{-4}$，飞行器质心偏离量的平均值与方差都很小，因此验证了问题 4 算法的有效性；算法的时间复杂度为$O(n^3)$。

8. 灵敏度分析

为保证求解的可靠性，本文在模型求解中引入了弃油量δ的上限值。由于上限值的设定会影响目标函数的最优解，如何选取合适的δ值是一个值得思考的问题，因此对弃油量的上限值做灵敏度分析。

取δ的数量级为 10^{-3}、10^{-2}、10^{-1}、0，采用问题 2 的数据得到不同δ时的最大质心偏移量如表 8. 1 所示，图 8. 1 为最大质心偏移量随δ数量级变化的曲线。

表 8. 1　不同δ时的最大质心偏移量

δ数量级	0	10^{-3}	10^{-2}	10^{-1}
最大质心偏移量(m)	0.022009233	0.017566159	0.021313734	0.746232157

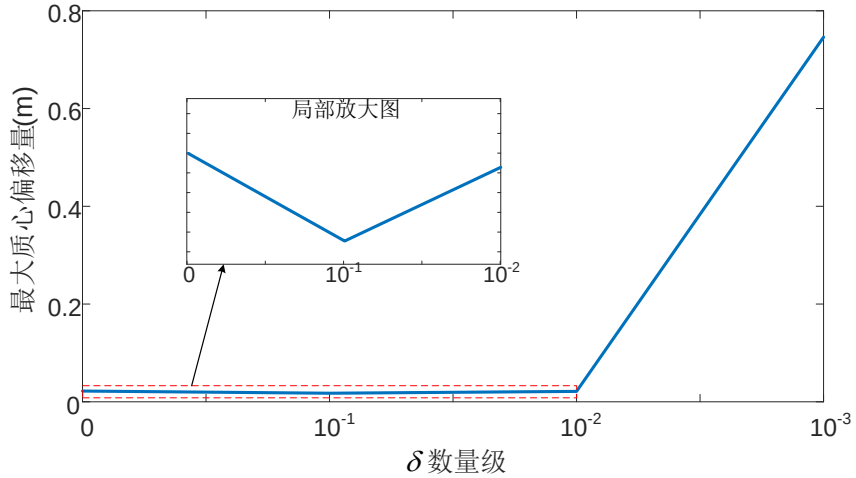


图 8.1　最大质心偏移量随δ数量级变化的曲线

从表 8. 1 和图 8. 1 可以得到：δ取数量级为 10^{-3} 时，最大质心偏移量最小。从图 8. 2 中可以得到，δ取数量级为 10^{-3} 时，其质心偏移量整体效果比数量级为 0 时优越，结合图 8. 3 可以得到，δ取数量级为 10^{-1} 时，飞行器前期的质心偏移量比数量级为 10^{-3} 的小，但其飞行器后期的质心偏移量会一直变大，这是因为前期会通过弃油来调节质心平衡且弃油量上限较大，后续飞行中不足以提供足够油量去调节质心平衡。

综上所述，本文取δ数量级为 10^{-3}，即$\delta = 5 \times 10^{-3}$。

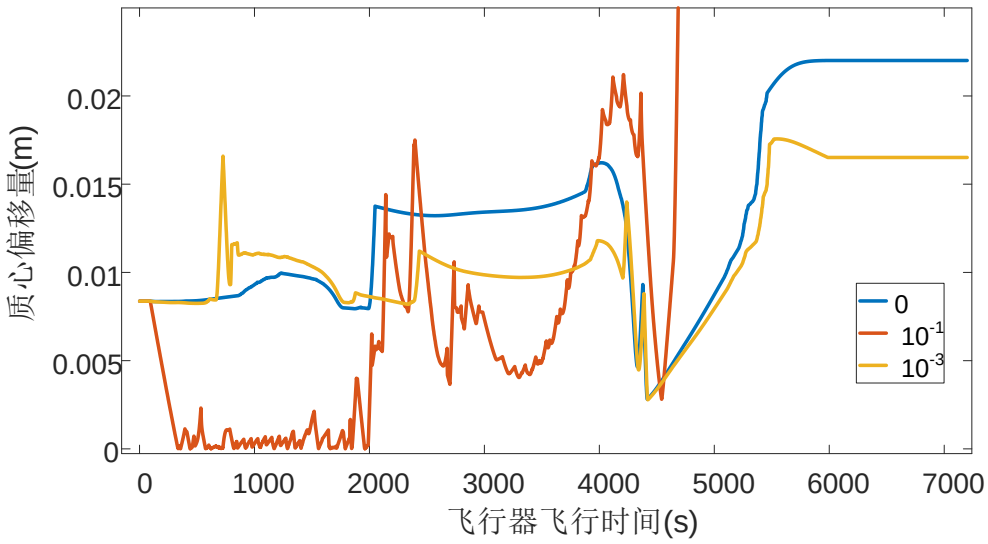


图 8.2　δ取数量级 10^{-3}、10^{-1}、0 每一时刻的质心偏移量曲线

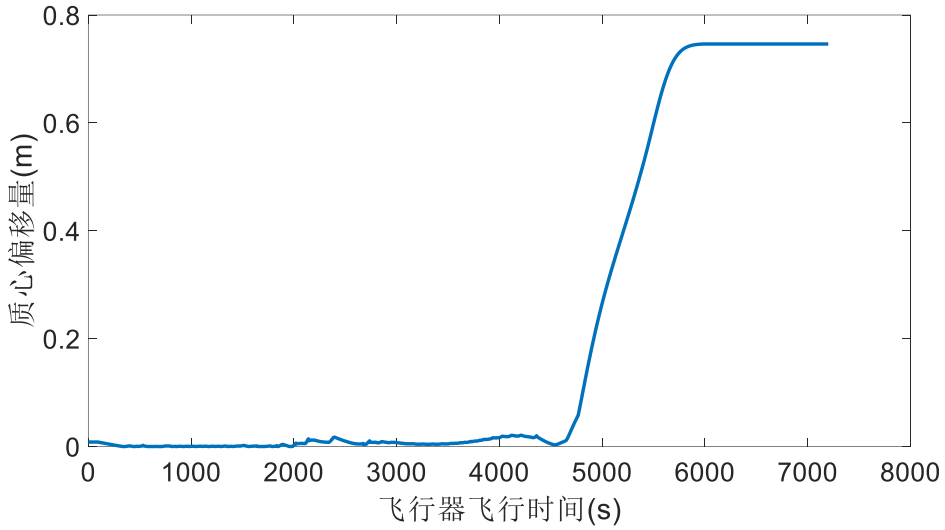


图 8.3　δ取数量级 10^{-1} 每一时刻的质心偏移量曲线

9. 模型评价与改进

9.1 模型评价

9.1.1 模型优点

（1）本文在建立供油策略优化模型时，考虑弃油量的上限值δ，降低了求解难度。且在实际中，节约燃油成本，保护环境。

（2）本文在建立供油策略优化模型时，引入油量最小限值ε（ε很小，可能小于挂在油箱壁的油量），降低模型求解难度，加快求解时间。并且油箱实际排油时会存在少量油留在油箱壁上，所以该限值有实际意义。

9.1.2 模型缺点

（1）处理连续的目标函数时，遗传算法有可能收敛于局部最优，算法有待改进。

（2）本文建立的数学模型是混合整数非线性模型，求解难度较大，模型有待改进。

9.2 模型改进

（1）在本文灵敏度分析（第 8 章）中，δ取数量级为 10^{-1} 的数值时，在飞行器飞行前段时间，其瞬时质心位置几乎与理想质心位置重合，而在飞行后期，由于油量不足且分布不均衡，本文算法无法继续优化质心位置，飞行器质心偏移量迅速增加（见图 8.3）。后续的算法改进可以添加记忆因子，若算法求解陷入无解状态，则跳回τ_i秒之前重新优化[6]。

（2）本文建立的数学模型属于混合整数非线性模型，求解难度较大。为了降低求解难度，本文提出数学模型改进建议，将非线性模型转化为部分线性模型，即将欧氏距离表达式线性化。欧式距离可以看成一个半径不定的球体，可以用 n 个多面体替代球体，将欧式距离表达式线性化[7]。

参考文献

[1]常胜利.多边形重心坐标的求法[J].高等数学研究,(02):21-23,2005.

[2]徐明波,冯琳娜.带约束的遗传算法理论研究及其在某型飞机耳片优化设计中的应用[J].科学技术与工程,11(32):7967-7971+8003,2011.

[3]谢晓锋,张文俊,阮骏,杨之廉.针对带约束非线性规划问题的遗传算法[J].计算机工程与应用,(21):64-67,2002.

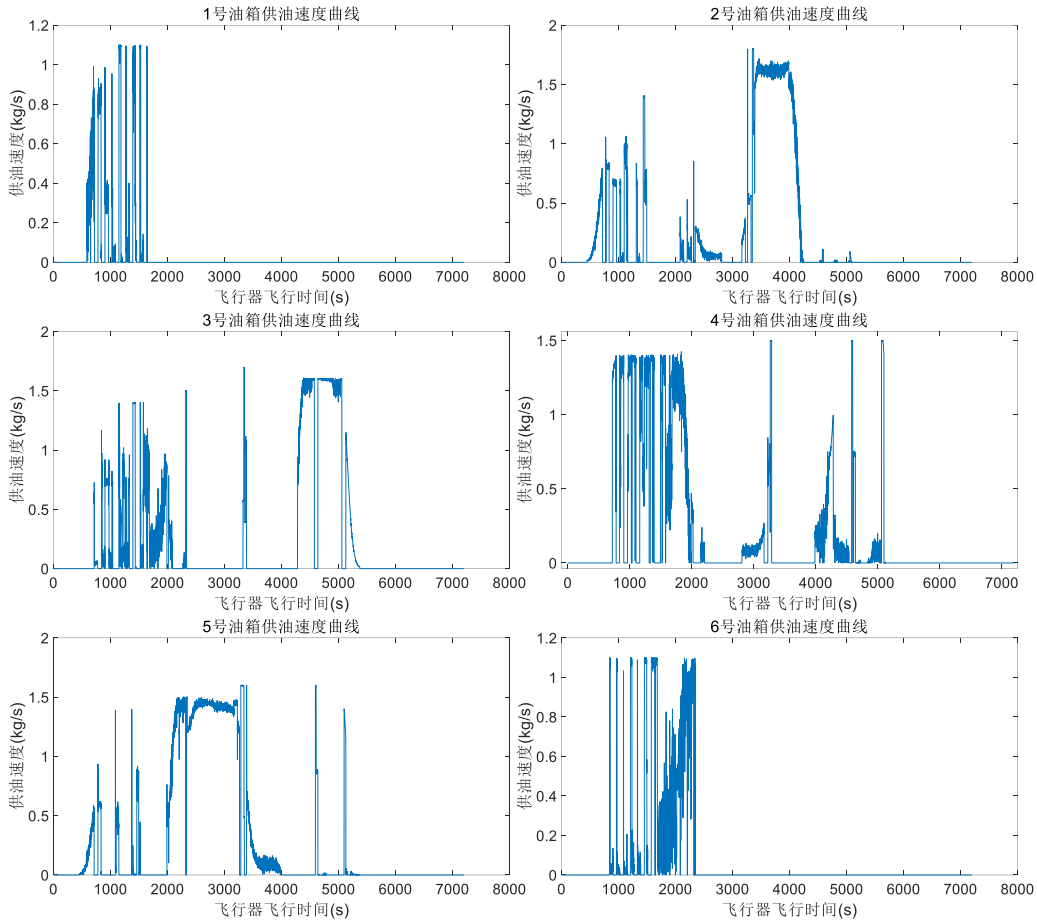
[4]张伟,黄卫民.基于种群分区的多策略自适应多目标粒子群算法[J/OL].自动化学报:1-14,2020-09-20.

[5]韦智勇,杨晓武.基于分层优化的资源受限系统任务拆分调度方法[J].计算机应用与软件,36(06):264-270,2019.

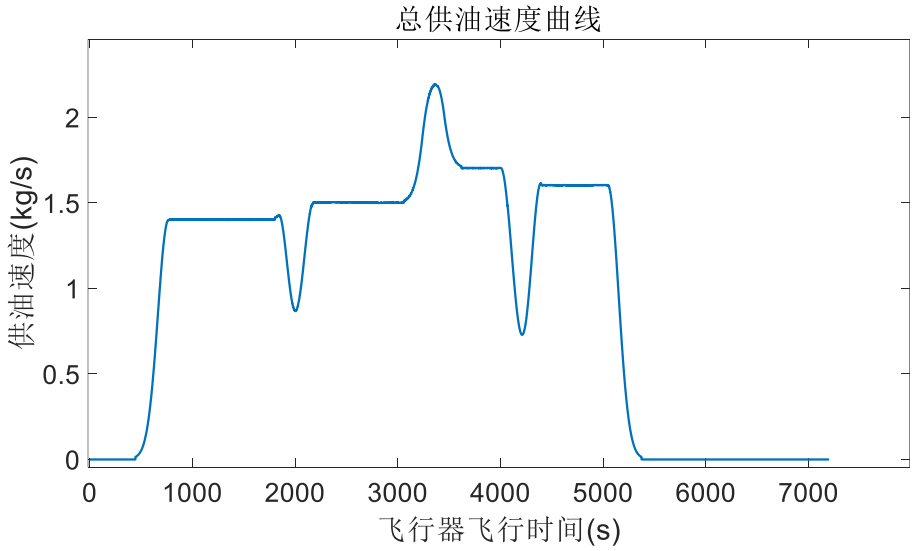
[6]何昕杰,周少武,张红强,吴亮红,周游.基于改进遗传算法的四向穿梭车系统订单排序优化[J/OL].系统仿真学报:1-14,2020-09-20.

[7]曲衍明.非凸二次约束优化问题的凸性化研究[D].北京邮电大学,2019.

附录 A 问题 3 的全局优化结果



附图 A.1　全局优化模型求解的 6 个油箱供油速度曲线



附图 A.2　全局优化模型求解的 4 个主油箱总供油速度曲线

附录 B 程序

问题 1：

```
#coding:utf-8
import math
import pandas as pd
def get_centerpoint(lis):     #需要按照多边形顺时或逆时针方向输入坐标点
    area = 0.0
    x,y = 0.0,0.0
    a = len(lis)
    for i in range(a):
        lat = lis[i][0] #weidu
        lng = lis[i][1] #jingdu
        if i == 0:
            lat1 = lis[-1][0]
            lng1 = lis[-1][1]
        else:
            lat1 = lis[i-1][0]
            lng1 = lis[i-1][1]
        fg = (lat*lng1 - lng*lat1)/2.0
        area += fg
        x += fg*(lat+lat1)/3.0
        y += fg*(lng+lng1)/3.0
    x = x/area
    y = y/area
    return x,y
def return_zx(x,y,z,zx_x,zx_y,zx_z,m):          #xyz 油箱的长宽高，zx_x,zx_y,zx_z 油箱中心坐标，m 油的质量
    temp_z=m/850.0/x/y
    z_temp=zx_z-z/2.0+temp_z/2.0
    return [zx_x,zx_y,z_temp]
def return_zx_dalt(x,y,z,zx_x,zx_y,zx_z,m,dalt):       #dalt  俯仰角度
    a=[]
    flag=0                #判断多边形的形状
    dm_s=m/850.0/y
    #tempx,y,z 为最底下的  参考点坐标
    tempz=zx_z-z/2.0
    tempy=zx_y
    tempx=zx_x-x/2.0
    #第一种情况：判断是否为三棱柱
    dbx_x=math.sqrt(2*dm_s/math.tan(abs(dalt)*math.pi/180.0))
    dbx_z=dbx_x*abs(math.tan(abs(dalt)*math.pi/180.0))
    if((dbx_x<=x) and (dbx_z<=z)):
        if(dalt>0):
            a.append([tempx,tempz])
```

```
                a.append([tempx,tempz+dbx_z])
                a.append([tempx+dbx_x,tempz])
            else:
                a.append([tempx+x,tempz])
                a.append([tempx+x,tempz+dbx_z])
                a.append([tempx+x-dbx_x,tempz])
            flag=1
        #第三种情况：判断是否为 五棱柱
        if (flag==0):
            dm_s1=x*z-dm_s
            dbx_x=math.sqrt(2*dm_s1/math.tan(abs(dalt)*math.pi/180.0))
            dbx_z=dbx_x*math.tan(abs(dalt)*math.pi/180.0)
            if((dbx_x<=x) and (dbx_z<=z)):
                if(dalt>0):
                    a.append([tempx,tempz])
                    a.append([tempx+x,tempz])
                    a.append([tempx+x,tempz+z-dbx_z])
                    a.append([tempx+x-dbx_x,tempz+z])
                    a.append([tempx,tempz+z])
                else:
                    a.append([tempx,tempz])
                    a.append([tempx,tempz+z-dbx_z])
                    a.append([tempx+dbx_x,tempz+z])
                    a.append([tempx+x,tempz+z])
                    a.append([tempx+x,tempz])
                flag=3
        #第二种情况：四棱柱
        if (flag==0):
            dbx_z=(dm_s-0.5*x*x*abs(math.tan(abs(dalt)*math.pi/180.0)))/x
            a.append([tempx,tempz])
            a.append([tempx+x,tempz])
            if(dalt<0):
                a.append([tempx+x,tempz+dbx_z+x*abs(math.tan(abs(dalt)*math.pi/180.0))])
                a.append([tempx,tempz+dbx_z])
            else:
                a.append([tempx+x,tempz+dbx_z])
                a.append([tempx,tempz+dbx_z+x*abs(math.tan(abs(dalt)*math.pi/180.0))])
            flag=2
        xx,zz=get_centerpoint(a)
        return [xx,tempy,zz]
XX1=pd.read_excel("附件 2-问题 1 数据.xlsx",sheet_name="油箱供油曲线")
XX2=pd.read_excel("附件 2-问题 1 数据.xlsx",sheet_name="飞行器俯仰角")
p_ranliao=850     #燃料密度
m_fj=3000         #无燃料飞机质量
```

```
#读取供油曲线(kg/s)
ft1=XX1["1 号油箱(kg/s)"].values
ft2=XX1["2 号油箱(kg/s)"].values
ft3=XX1["3 号油箱(kg/s)"].values
ft4=XX1["4 号油箱(kg/s)"].values
ft5=XX1["5 号油箱(kg/s)"].values
ft6=XX1["6 号油箱(kg/s)"].values
#读取俯仰角度变化
f_dalt=XX2["飞行器俯仰角(度)"].values
#油箱长宽高参数
yx_x=[1.5,2.2,2.4,1.7,2.2,2.4]   #长
yx_y=[0.9,0.8,1.1,1.3,1.2,1]     #宽
yx_z=[0.3,1.1,0.9,1.2,1,0.5]     #高
#油箱中心坐标
yz_x=[8.91304348,6.91304348,-1.68695652,3.11304348,-5.28695652,-2.08695652]
yz_y=[1.20652174,-1.39347826,1.20652174,0.60652174,-0.29347826,-1.49347826]
yz_z=[0.61669004,0.21669004,-0.28330996,-0.18330996,0.41669004,0.21669004]
#油箱初始油量
yc=[0.3,1.5,2.1,1.9,2.6,0.8]
yc_m=[]              #初始油的质量
for i in range(len(yc)):
    yc_m.append(yc[i]*p_ranliao)
#  质心 x,y,z
Z_x=[]
Z_y=[]
Z_z=[]
for i in range(0,len(f_dalt)):
    #计算各个油箱的油量
    xyz_zb=[]              #每个油箱的重心
    yxz_zx=[]              #油箱系统的重心
    yc_m[0]-=ft1[i];yc_m[2]-=ft3[i]
    yc_m[3]-=ft4[i];yc_m[5]-=ft6[i]
    yc_m[1]-=ft2[i];yc_m[4]-=ft5[i]
    yc_m[1]+=ft1[i];yc_m[4]+=ft6[i]
    sum_m=yc_m[0]+yc_m[1]+yc_m[2]+yc_m[3]+yc_m[4]+yc_m[5]          #油总质量
    if(f_dalt[i]==0):
        for j in range(0,len(yc_m)):
            xyz_zb.append(return_zx(yx_x[j],yx_y[j],yx_z[j],yz_x[j],yz_y[j],yz_z[j],yc_m[j]))
        x1=y1=z1=0
        for j in range(0,len(xyz_zb)):
            x1+=xyz_zb[j][0]*yc_m[j]
            y1+=xyz_zb[j][1]*yc_m[j]
            z1+=xyz_zb[j][2]*yc_m[j]
        yxz_zx.append(x1/(sum_m+m_fj))
```

```
            yxz_zx.append(y1/(sum_m+m_fj))
            yxz_zx.append(z1/(sum_m+m_fj))
        else:
            for j in range(0,len(yc_m)):
xyz_zb.append(return_zx_dalt(yx_x[j],yx_y[j],yx_z[j],yz_x[j],yz_y[j],yz_z[j],yc_m[j],f_dalt[i]))
            x1=y1=z1=0
            for j in range(0,len(xyz_zb)):
                x1+=xyz_zb[j][0]*yc_m[j]
                y1+=xyz_zb[j][1]*yc_m[j]
                z1+=xyz_zb[j][2]*yc_m[j]
            yxz_zx.append(x1/(sum_m+m_fj))
            yxz_zx.append(y1/(sum_m+m_fj))
            yxz_zx.append(z1/(sum_m+m_fj))
        Z_x.append(yxz_zx[0])
        Z_y.append(yxz_zx[1])
        Z_z.append(yxz_zx[2])
XX3=pd.DataFrame()
XX3["质心 x 坐标"]=Z_x
XX3["质心 y 坐标"]=Z_y
XX3["质心 z 坐标"]=Z_z
XX3.to_excel("重心坐标.xlsx")
```

问题 2：（遗传算法）

```
#coding :utf-8
import random
import pandas as pd
import math
import numpy as np
#数据的读取
XX1=pd.read_excel("附件 3-问题 2 数据.xlsx", sheet_name="发动机耗油速度")
XX2=pd.read_excel("附件 3-问题 2 数据.xlsx", sheet_name="飞行器理想质心数据")
YH=XX1["耗油速度(kg/s)"].values
CK_X=XX2["X 坐标（米）"].values
CK_Y=XX2["y 坐标（米）"].values
CK_Z=XX2["z 坐标（米）"].values
p_ranliao=850    #燃料密度
m_fj=3000        #无燃料飞机质量
#油箱长宽高参数
yx_x=[1.5, 2.2, 2.4, 1.7, 2.2, 2.4]   #长
yx_y=[0.9, 0.8, 1.1, 1.3, 1.2, 1]     #宽
yx_z=[0.3, 1.1, 0.9, 1.2, 1, 0.5]     #高
#油箱中心坐标
yz_x=[8.91304348, 6.91304348, -1.68695652, 3.11304348, -5.28695652, -2.08695652]
yz_y=[1.20652174, -1.39347826, 1.20652174, 0.60652174, -0.29347826, -1.49347826]
```

```
yz_z=[0.61669004, 0.21669004, -0.28330996, -0.18330996, 0.41669004, 0.21669004]
#油箱初始油量
yc=[0.3, 1.5, 2.1, 1.9, 2.6, 0.8]
yc_m=[]              #初始油的质量
for i in range(len(yc)):
    yc_m.append(yc[i]*p_ranliao)
def return_zx(x, y, z, zx_x, zx_y, zx_z, m):          #xyz 油箱的长宽高，zx_x,zx_y,zx_z 油箱中心坐标，m 油的质量
    temp_z=m/850.0/x/y
    z_temp=zx_z-z/2.0+temp_z/2.0
    return [zx_x, zx_y, z_temp]
def return_zx_sum(x, y, z, zx_x, zx_y, zx_z, m):          #返回飞机质心坐标
    xyz_zb=[]
    yxz_zx=[]
    sum_m=0
    for i in range(0, len(m)):
        sum_m+=m[i]
    for j in range(0, len(m)):
        xyz_zb.append(return_zx(x[j], y[j], z[j], zx_x[j], zx_y[j], zx_z[j], m[j]))
    x1=y1=z1=0
    for j in range(0, len(xyz_zb)):
        x1+=xyz_zb[j][0]*m[j]
        y1+=xyz_zb[j][1]*m[j]
        z1+=xyz_zb[j][2]*m[j]
    yxz_zx.append(x1/(sum_m+m_fj))
    yxz_zx.append(y1/(sum_m+m_fj))
    yxz_zx.append(z1/(sum_m+m_fj))
    return yxz_zx
def chanshen_bianliang(YH, mi, ti):          #返回变量值—基于遗传
    t=1.0
    m_t=0.1          #油量限制
    py_l=5e-2        #排油限制
    ub=[1.1*t, 1.8*t, 1.7*t, 1.5*t, 1.6*t, 1.1*t]
    while(1):
        xi=[0, 0, 0, 0, 0, 0]
        flag=1
        ei=[0, 0, 0, 0, 0, 0]
        ai=[0, 5]
        if(mi[0]<m_t):ai.remove(0)
        elif(ti[0]>=1 and ti[0]<=60):ai.remove(0);ei[0]=1;
        if(mi[5]<m_t):ai.remove(5)
        elif(ti[5]>=1 and ti[5]<=60):ai.remove(5);ei[5]=1;
        aj=[1, 2, 3, 4]
        if(mi[1]<m_t):aj.remove(1)
        elif(ti[1]>=1 and ti[1]<=60):aj.remove(1);ei[1]=1;
```

```
        if(mi[2]<m_t):aj.remove(2)
        elif(ti[2]>=1 and ti[2]<=60):aj.remove(2);ei[2]=1;
        if(mi[3]<m_t):aj.remove(3)
        elif(ti[3]>=1 and ti[3]<=60):aj.remove(3);ei[3]=1;
        if(mi[4]<m_t):aj.remove(4)
        elif(ti[4]>=1 and ti[4]<=60):aj.remove(4);ei[4]=1;
        sum_gy=0;sum_zg=0
        for i in range(1,5):
            if(ei[i]==1):
                sum_gy+=1
        for i in [0,5]:
            if(ei[i]==1):
                sum_zg+=1
        if(sum_gy==0):
            if(YH<ub[1] and YH<ub[2]and YH<ub[3]and YH<ub[4]):
                if(YH!=0):shenyu1=random.randint(1,2)
                else:shenyu1=0
            else:shenyu1=2
        elif(sum_gy==1):
            if(YH<ub[1] and YH<ub[2]and YH<ub[3]and YH<ub[4]):shenyu1=random.randint(0,1)
            else:shenyu1=1
        elif(sum_gy==2):shenyu1=0
        if(shenyu1>len(aj)):shenyu1=len(aj)
        if(shenyu1!=0):
            for i in range(0,shenyu1):
                a=random.choice(aj)
                aj.remove(a)
                ei[a]=1
        if(sum_zg==0):shenyu2=random.randint(0,1)
        else:shenyu2=0
        if(shenyu2>len(ai)):shenyu2=len(ai)
        if(shenyu2!=0 and len(ai)!=0):
            for i in range(0,shenyu2):
                if(mi[0]<=m_t or mi[5]<=m_t):
                    if(mi[5]>m_t or mi[0]>m_t):
                        if (mi[0]<m_t):
                            ei[5]=1
                        elif(mi[5]<m_t):
                            ei[0]=1
                    else:
                        break
                else:
                    a=random.choice(ai)
                    ai.remove(a)
```

```
                    ei[a]=1
        flag_1=[]
        flag_2=[]
        flag_3=0#跳出循环标志
        for i in range(0, len(ei)):
            if(i==0 or i==5):
                if(ei[i]==1):flag_2.append(i)
            else:
                if(ei[i]==1):flag_1.append(i)
        if (len(flag_1)==0):pass
        elif(len(flag_1)==1):
            if(mi[flag_1[0]]>=ub[flag_1[0]] and YH<mi[flag_1[0]]):
                xi[flag_1[0]]=random.uniform(YH, YH+py_l)
        elif(len(flag_1)==2):
            if(YH!=0):
                while(xi[flag_1[0]]<=0 or xi[flag_1[0]]>=ub[flag_1[0]] or xi[flag_1[1]]<=0 or
xi[flag_1[1]]>=ub[flag_1[1]]):
                    if(mi[flag_1[0]]>YH):
                        xi[flag_1[0]]=random.uniform(0, YH+py_l)
                        if(flag_3>=500):
                            ti[flag_1[1]]=0
                    else:
                        xi[flag_1[0]]=random.uniform(0, mi[flag_1[0]])
                        if(flag_3>=500):
                            ti[flag_1[0]]=0
                    xi[flag_1[1]]=random.uniform(YH, YH+py_l)-xi[flag_1[0]]
                    flag_3+=1;
                    if(flag_3>=500):break
        if(flag_3>=500):continue
        for m in flag_2:
            if(mi[m]>ub[m]):
                xi[m]=random.uniform(0, ub[m])
            else:
                xi[m]=random.uniform(0, mi[m])
        for i in range(0, len(mi)):
            if(mi[i]-xi[i]>=0):flag=flag&1
            else:flag=flag&0
        if(flag==1):
            break
    return xi
ti=[0, 0, 0, 0, 0, 0]
xi=[0, 0, 0, 0, 0, 0]
FAn_60s=[]          #供油变化量
YH_sum=[]           #油耗和
```

```
OBJ=[]
t=1
for k in range(0, int(len(YH)/t)):
    yh_SUM=0
    obj=100
    for j in range(0, t):
        yh_SUM+=YH[t*k+j]
    print(yh_SUM)
    YH_sum.append(yh_SUM)
    for j in range(0, 1000):
        m_temp=[]
        x_temp=chanshen_bianliang(yh_SUM, yc_m, ti)
        for i in range(0, len(x_temp)):
            m_temp.append(yc_m[i]-x_temp[i])
            if(i==1):
                m_temp[1]+=x_temp[0]
            if(i==4):
                m_temp[4]+=x_temp[5]
        zx=return_zx_sum(yx_x, yx_y, yx_z, yz_x, yz_y, yz_z, m_temp) #求取质心
        obj_temp=math.sqrt((CK_X[t*k]-zx[0])**2+(CK_Y[t*k]-zx[1])**2+(CK_Z[t*k]-zx[2])**2)
        if(obj_temp<obj):
            obj=obj_temp
            out_zx=[];out_x=[];out_m=[]
            for m in  zx:
                out_zx.append(m)
            for m in x_temp:
                out_x.append(m)
            for m in m_temp:
                out_m.append(m)
        del zx, x_temp, m_temp
    yc_m=[]
    for m in out_m:
        yc_m.append(m)
    print(k)
    for i in range(0, len(out_x)):
        if(out_x[i]!=0):
            ti[i]+=1
        else:
            ti[i]=0
    print(out_zx); print(obj); print(out_x)
    print(out_m); print(ti)
    FAn_60s.append(out_x)
    OBJ.append(obj)
yx1=[];yx2=[];yx3=[];yx4=[];yx5=[];yx6=[]
```

```
for z in FAn_60s:
    yx1.append(z[0]);yx2.append(z[1]); yx3.append(z[2])
    yx4.append(z[3]); yx5.append(z[4]); yx6.append(z[5])
XX3=pd.DataFrame()
XX3["1 号"]=yx1;XX3["2 号"]=yx2;XX3["3 号"]=yx3
XX3["4 号"]=yx4;XX3["5 号"]=yx5;XX3["6 号"]=yx6
XX3["目标"]=OBJ
print("最大偏差：",np.max(np.array(OBJ)))
XX3.to_excel("供油曲线 2_10.xlsx")
```

问题 2：（粒子群算法）

```
#coding:utf-8
import random
import pandas as pd
import math
import numpy as np
from sko.PSO import PSO
#数据的读取
XX1=pd.read_excel("附件 3-问题 2 数据.xlsx",sheet_name="发动机耗油速度")
XX2=pd.read_excel("附件 3-问题 2 数据.xlsx",sheet_name="飞行器理想质心数据")
YH=XX1["耗油速度(kg/s)"].values
CK_X=XX2["X 坐标（米）"].values
CK_Y=XX2["y 坐标（米）"].values
CK_Z=XX2["z 坐标（米）"].values
p_ranliao=850    #燃料密度
m_fj=3000        #无燃料飞机质量
#油箱长宽高参数
yx_x=[1.5,2.2,2.4,1.7,2.2,2.4]   #长
yx_y=[0.9,0.8,1.1,1.3,1.2,1]     #宽
yx_z=[0.3,1.1,0.9,1.2,1,0.5]     #高
#油箱中心坐标
yz_x=[8.91304348,6.91304348,-1.68695652,3.11304348,-5.28695652,-2.08695652]
yz_y=[1.20652174,-1.39347826,1.20652174,0.60652174,-0.29347826,-1.49347826]
yz_z=[0.61669004,0.21669004,-0.28330996,-0.18330996,0.41669004,0.21669004]
#油箱初始油量
yc=[0.3,1.5,2.1,1.9,2.6,0.8]
yc_m=[]          #初始油的质量
for i in range(len(yc)):
    yc_m.append(yc[i]*p_ranliao)
def return_zx(x,y,z,zx_x,zx_y,zx_z,m):        #xyz 油箱的长宽高，zx_x,zx_y,zx_z 油箱中心坐标，m 油的质量
    temp_z=m/850.0/x/y
    z_temp=zx_z-z/2.0+temp_z/2.0
    return [zx_x,zx_y,z_temp]
def return_zx_sum(x,y,z,zx_x,zx_y,zx_z,m):    #返回飞机质心坐标
```

```
    xyz_zb=[];yxz_zx=[]
    sum_m=0
    for i in range(0, len(m)):
        sum_m+=m[i]
    for j in range(0, len(m)):
        xyz_zb.append(return_zx(x[j], y[j], z[j], zx_x[j], zx_y[j], zx_z[j], m[j]))
    x1=y1=z1=0
    for j in range(0, len(xyz_zb)):
        x1+=xyz_zb[j][0]*m[j]
        y1+=xyz_zb[j][1]*m[j]
        z1+=xyz_zb[j][2]*m[j]
    yxz_zx.append(x1/(sum_m+m_fj));  yxz_zx.append(y1/(sum_m+m_fj));yxz_zx.append(z1/(sum_m+m_fj))
    return yxz_zx
ti=[0, 0, 0, 0, 0, 0]              #0-1 变量
m_sum=[0, 0, 0, 0, 0, 0]           #剩余油量
XU_qiu=0
ZX1=[0, 0, 0]
def demo_func(x):
    x1,  x2,  x3, x4, x5, x6= x
    X=[x1, x2, x3, x4, x5, x6]
    Ti=[0, 0, 0, 0, 0, 0]              #供油 60s
    Ti_1=[0, 0, 0, 0, 0, 0]
    SUM=x2+x3+x4+x5
    for i in range(0, len(yc_m)):              #油箱油量大于 0
        m_sum[i]=yc_m[i]-X[i]
    for m in m_sum:
        if(m<0):return 99
    for i in range(0, len(X)):            #记此时刻的状态
        if(X[i]!=0):
            Ti_1[i]=1
    for i in range(0, len(ti)):
        if(ti[i]>0 and ti[i]<=60):          #约束：供油不少于 60s
            Ti[i]=1
            Ti_1[i]=1
    for i in range(0, len(Ti)):
        if(Ti[i]!=0):
            if(X[i]==0):
                return 99
    sum_gy=Ti_1[1]+Ti_1[2]+Ti_1[3]+Ti_1[4]
    sum_zg=Ti_1[0]+Ti_1[1]+Ti_1[2]+Ti_1[3]+Ti_1[4]+Ti_1[5]
    if(sum_gy>2):return 99          #约束：供油不超过两个
    if(sum_zg>3):return 99          #约束：同时不超过三个
    if(SUM<XU_qiu or SUM>XU_qiu+0.005):
        return 99
```

```
    else:
        ZX=return_zx_sum(yx_x, yx_y, yx_z, yz_x, yz_y, yz_z, m_sum)
        return math.sqrt((ZX1[0]-ZX[0])**2+(ZX1[1]-ZX[1])**2+(ZX1[2]-ZX[2])**2)


Yx=0.1
lb1=[0, 0, 0, 0, 0, 0]
ub1=[1.1, 1.8, 1.7, 1.5, 1.6, 1.1]
FAn_60s=[]          #供油变化量
OBJ=[]
for z in range(0, len(YH)):
    flag_1=0
    XU_qiu=YH[z]
    ZX1=[CK_X[z], CK_Y[z], CK_Z[z]]
    print("time:", z)
    for i in range(0, len(yc_m)):            #供油下限：小于限制则认为油箱不供油
        if(yc_m[i]<Yx):
            ti[i]=-1            #油箱退出供油状态
    while(1):
        pso = PSO(func=demo_func, dim=6, pop=2000, max_iter=5, lb=lb1, ub=ub1, w=1, c1=0.9, c2=0.9)
        pso.run()
        print('best_x is ', pso.gbest_x, 'best_y is', pso.gbest_y)
        if(pso.gbest_y==99):flag_1+=1;continue
        else:break
        if(flag_1==1000):print("无解！");sys.exit(0);
    X_temp=list(pso.gbest_x)
    for i in range(0, len(yc_m)):
        yc_m[i]=yc_m[i]-X_temp[i]
    print("剩余量：", yc_m)
    for i in range(0, len(X_temp)):
        if(X_temp[i]!=0):
            ti[i]+=1
        else:
            ti[i]=0
    print("供油时间：", ti)
    FAn_60s.append(X_temp)
    OBJ.append(pso.gbest_y)

yx1=[];yx2=[];yx3=[];yx4=[];yx5=[];yx6=[]
for z in FAn_60s:
    yx1.append(z[0]);yx2.append(z[1]); yx3.append(z[2])
    yx4.append(z[3]); yx5.append(z[4]); yx6.append(z[5])
XX3=pd.DataFrame()
XX3["1号"]=yx1;XX3["2号"]=yx2;XX3["3号"]=yx3
XX3["4号"]=yx4;XX3["5号"]=yx5;XX3["6号"]=yx6
```

```
XX3["目标"]=OBJ
print("最大偏差：", np.max(np.array(OBJ)))
XX3.to_excel("供油曲线 2_10.xlsx")
```

问题 3：

```
#coding:utf-8
import random
import pandas as pd
import math
import numpy as np
#数据的读取
XX1=pd.read_excel("附件 4-问题 3 数据.xlsx", sheet_name="发动机耗油数据")
XX2=pd.read_excel("附件 4-问题 3 数据.xlsx", sheet_name="飞行器理想质心")
YH=XX1["耗油速度(kg/s)"].values
CK_X=XX2["X 坐标（米）"].values
CK_Y=XX2["y 坐标（米）"].values
CK_Z=XX2["z 坐标（米）"].values
p_ranliao=850    #燃料密度
m_fj=3000        #无燃料飞机质量
#油箱长宽高参数
yx_x=[1.5, 2.2, 2.4, 1.7, 2.2, 2.4]  #长
yx_y=[0.9, 0.8, 1.1, 1.3, 1.2, 1]    #宽
yx_z=[0.3, 1.1, 0.9, 1.2, 1, 0.5]    #高
#油箱中心坐标
yz_x=[8.91304348, 6.91304348, -1.68695652, 3.11304348, -5.28695652, -2.08695652]
yz_y=[1.20652174, -1.39347826, 1.20652174, 0.60652174, -0.29347826, -1.49347826]
yz_z=[0.61669004, 0.21669004, -0.28330996, -0.18330996, 0.41669004, 0.21669004]
#油箱初始油量
# yc=[0.3,1.5,2.1,1.9,2.6,0.8]
def return_zx(x, y, z, zx_x, zx_y, zx_z, m):        #xyz 油箱的长宽高，zx_x,zx_y,zx_z 油箱中心坐标，m 油的质量
    temp_z=m/850.0/x/y
    z_temp=zx_z-z/2.0+temp_z/2.0
    return [zx_x, zx_y, z_temp]
def return_zx_sum(x, y, z, zx_x, zx_y, zx_z, m):        #返回飞机质心坐标
    xyz_zb=[]
    yxz_zx=[]
    sum_m=0
    for i in range(0, len(m)):
        sum_m+=m[i]
    for j in range(0, len(m)):
        xyz_zb.append(return_zx(x[j], y[j], z[j], zx_x[j], zx_y[j], zx_z[j], m[j]))
    x1=y1=z1=0
    for j in range(0, len(xyz_zb)):
        x1+=xyz_zb[j][0]*m[j]
```

```
            y1+=xyz_zb[j][1]*m[j]
            z1+=xyz_zb[j][2]*m[j]
        yxz_zx.append(x1/(sum_m+m_fj))
        yxz_zx.append(y1/(sum_m+m_fj))
        yxz_zx.append(z1/(sum_m+m_fj))
        return yxz_zx
def chanshen_bianliang(YH, mi, ti):          #返回变量值一基于遗传
    t=1.0
    m_t=1e-1          #油量限制
    py_l=0.005      #排油限制
    ub=[1.1*t, 1.8*t, 1.7*t, 1.5*t, 1.6*t, 1.1*t]
    while(1):
        xi=[0, 0, 0, 0, 0, 0]
        flag=1
        ei=[0, 0, 0, 0, 0, 0]
        ai=[0, 5]
        if(mi[0]<m_t):ai.remove(0)
        elif(ti[0]>=1 and ti[0]<=60):ai.remove(0);ei[0]=1;
        if(mi[5]<m_t):ai.remove(5)
        elif(ti[5]>=1 and ti[5]<=60):ai.remove(5);ei[5]=1;
        aj=[1, 2, 3, 4]
        if(mi[1]<m_t):aj.remove(1)
        elif(ti[1]>=1 and ti[1]<=60):aj.remove(1);ei[1]=1;
        if(mi[2]<m_t):aj.remove(2)
        elif(ti[2]>=1 and ti[2]<=60):aj.remove(2);ei[2]=1;
        if(mi[3]<m_t):aj.remove(3)
        elif(ti[3]>=1 and ti[3]<=60):aj.remove(3);ei[3]=1;
        if(mi[4]<m_t):aj.remove(4)
        elif(ti[4]>=1 and ti[4]<=60):aj.remove(4);ei[4]=1;
        sum_gy=0;sum_zg=0
        for i in range(1, 5):
            if(ei[i]==1):
                sum_gy+=1
        for i in [0, 5]:
            if(ei[i]==1):
                sum_zg+=1
        if(sum_gy==0):
            if(YH<ub[1] and YH<ub[2]and YH<ub[3]and YH<ub[4]):
                if(YH!=0):shenyu1=random.randint(1, 2)
                else:shenyu1=0
            else:shenyu1=2
        elif(sum_gy==1):
            if(YH<ub[1] and YH<ub[2]and YH<ub[3]and YH<ub[4]):shenyu1=random.randint(0, 1)
            else:shenyu1=1
```

```
        elif(sum_gy==2):shenyu1=0
        if(shenyu1>len(aj)):shenyu1=len(aj)
        if(shenyu1!=0):
            for i in range(0,shenyu1):
                a=random.choice(aj)
                aj.remove(a)
                ei[a]=1
        if(sum_zg==0):shenyu2=random.randint(0,1)
        else:shenyu2=0
        if(shenyu2>len(ai)):shenyu2=len(ai)
        if(shenyu2!=0 and len(ai)!=0):
            for i in range(0,shenyu2):
                if(mi[0]<=m_t or mi[5]<=m_t):
                    if(mi[5]>m_t or mi[0]>m_t):
                        if (mi[0]<m_t):
                            ei[5]=1
                        elif(mi[5]<m_t):
                            ei[0]=1
                    else:
                        break
                else:
                    a=random.choice(ai)
                    ai.remove(a)
                    ei[a]=1
        flag_1=[]
        flag_2=[]
        flag_3=0#跳出循环标志
        for i in range(0,len(ei)):
            if(i==0 or i==5):
                if(ei[i]==1):flag_2.append(i)
            else:
                if(ei[i]==1):flag_1.append(i)
        if (len(flag_1)==0):pass
        elif(len(flag_1)==1):
            if(mi[flag_1[0]]>=ub[flag_1[0]] and YH<mi[flag_1[0]]):
                xi[flag_1[0]]=random.uniform(YH,YH+py_l)
        elif(len(flag_1)==2):
            if(YH!=0):
                while(xi[flag_1[0]]<=0 or xi[flag_1[0]]>=ub[flag_1[0]] or xi[flag_1[1]]<=0 or
xi[flag_1[1]]>=ub[flag_1[1]]):
                    if(mi[flag_1[0]]>YH):
                        xi[flag_1[0]]=random.uniform(0,YH+py_l)
                        if(flag_3>=500):
                            ti[flag_1[1]]=0
```

```
                    else:
                        xi[flag_1[0]]=random.uniform(0, mi[flag_1[0]])
                        if(flag_3>=500):
                            ti[flag_1[0]]=0
                    xi[flag_1[1]]=random.uniform(YH, YH+py_l)-xi[flag_1[0]]
                    flag_3+=1;
                    if(flag_3>=500):break
            if(flag_3>=500):continue
            for m in flag_2:
                if(mi[m]>ub[m]):
                    xi[m]=random.uniform(0, ub[m])
                else:
                    xi[m]=random.uniform(0, mi[m])
            for i in range(0, len(mi)):
                if(mi[i]-xi[i]>=0):flag=flag&1
                else:flag=flag&0
            if(flag==1):
                break
        return xi
def you_liang(ZL, ub_l):
        X=[0, 0, 0, 0, 0, 0]
        ZL_1=ZL+random.uniform(0, 0.4)
        while(1):
            sum=0
            for i in range(0, len(X)-1):
                X[i]=random.uniform(0, ub_l[i])
                sum+=X[i]
            X[-1]=ZL_1-sum
            if(X[-1]<ub_l[-1] and X[-1]>=0):
                break
        return X
ub_l=[]
for i in range(0, len(yx_x)):
        ub_l.append(yx_x[i]*yx_y[i]*yx_z[i])
chushi_=[]
obj=99
for i in range(0, 10000):
        m=[]
        X=you_liang(9.01, ub_l)
        for j in range(0, len(X)):
            m.append(X[j]*850)
        zx=return_zx_sum(yx_x, yx_y, yx_z, yz_x, yz_y, yz_z, m)
        obj_temp=math.sqrt((CK_X[0]-zx[0])**2+(CK_Y[0]-zx[1])**2+(CK_Z[0]-zx[2])**2)
        if(obj_temp<obj):
```

```
            obj=obj_temp
            X_e=X
yc=X_e
yc_m=[]              #初始油的质量
for i in range(len(yc)):
    yc_m.append(yc[i]*p_ranliao)
ti=[0, 0, 0, 0, 0, 0]
xi=[0, 0, 0, 0, 0, 0]
FAn_60s=[]          #供油变化量
YH_sum=[]           #油耗和
OBJ=[]
t=1
for k in range(0, int(len(YH)/t)):
    yh_SUM=0
    obj=100
    for j in range(0, t):
        yh_SUM+=YH[t*k+j]
    print(yh_SUM)
    YH_sum.append(yh_SUM)
    for j in range(0, 1000):
        m_temp=[]
        x_temp=chanshen_bianliang(yh_SUM, yc_m, ti)
        for i in range(0, len(x_temp)):
            m_temp.append(yc_m[i]-x_temp[i])
            if(i==1):
                m_temp[1]+=x_temp[0]
            if(i==4):
                m_temp[4]+=x_temp[5]
        zx=return_zx_sum(yx_x, yx_y, yx_z, yz_x, yz_y, yz_z, m_temp)#求取质心
        obj_temp=math.sqrt((CK_X[t*k]-zx[0])**2+(CK_Y[t*k]-zx[1])**2+(CK_Z[t*k]-zx[2])**2)
        if(obj_temp<obj):
            obj=obj_temp
            out_zx=[];out_x=[];out_m=[]
            for m in  zx:
                out_zx.append(m)
            for m in x_temp:
                out_x.append(m)
            for m in m_temp:
                out_m.append(m)
        del zx, x_temp, m_temp
    yc_m=[]
    for m in out_m:
        yc_m.append(m)
    print(k)
```

```
    for i in range(0, len(out_x)):
        if(out_x[i]!=0):
            ti[i]+=1
        else:
            ti[i]=0
    print(out_zx); print(obj)
    print(out_x); print(out_m); print(ti)
    FAn_60s.append(out_x)
    OBJ.append(obj)
yx1=[];yx2=[];yx3=[];yx4=[];yx5=[];yx6=[]
for z in FAn_60s:
    yx1.append(z[0]); yx2.append(z[1]); yx3.append(z[2])
    yx4.append(z[3]); yx5.append(z[4]); yx6.append(z[5])
XX3=pd.DataFrame()
XX3["1 号"]=yx1;XX3["2 号"]=yx2;XX3["3 号"]=yx3
XX3["4 号"]=yx4;XX3["5 号"]=yx5;XX3["6 号"]=yx6
XX3["目标"]=OBJ
print("最大偏差：", np.max(np.array(OBJ)))
print("油量：", yc)
print("供油曲线 3_9.xlsx")
XX3.to_excel("供油曲线 3_9.xlsx")
```

问题 4：

```
#coding:utf-8
import random
import pandas as pd
import math
import numpy as np
p_ranliao=850    #燃料密度
m_fj=3000        #无燃料飞机质量
def get_centerpoint(lis):    #需要按照多边形顺时或逆时针方向输入坐标点
    area = 0.0
    x, y = 0.0, 0.0
    a = len(lis)
    for i in range(a):
        lat = lis[i][0] #weidu
        lng = lis[i][1] #jingdu
        if i == 0:
            lat1 = lis[-1][0]
            lng1 = lis[-1][1]
        else:
            lat1 = lis[i-1][0]
            lng1 = lis[i-1][1]
```

```
            fg = (lat*lng1 - lng*lat1)/2.0
            area += fg
            x += fg*(lat+lat1)/3.0
            y += fg*(lng+lng1)/3.0
        x = x/area
        y = y/area
        return x, y
def return_zx(x, y, z, zx_x, zx_y, zx_z, m):          #xyz 油箱的长宽高，zx_x,zx_y,zx_z 油箱中心坐标，m 油的质量
        temp_z=m/850.0/x/y
        z_temp=zx_z-z/2.0+temp_z/2.0
        return [zx_x, zx_y, z_temp]
def return_zx_dalt(x, y, z, zx_x, zx_y, zx_z, m, dalt):      #dalt 俯仰角度
        a=[]
        flag=0              #判断多边形的形状
        dm_s=m/850.0/y
        #tempx,y,z 为最底下的 参考点坐标
        tempz=zx_z-z/2.0
        tempy=zx_y
        tempx=zx_x-x/2.0
        #第一种情况：判断是否为三棱柱
        dbx_x=math.sqrt(2*dm_s/math.tan(abs(dalt)*math.pi/180.0))
        dbx_z=dbx_x*abs(math.tan(abs(dalt)*math.pi/180.0))
        if((dbx_x<=x) and (dbx_z<=z)):
            if(dalt>0):
                a.append([tempx, tempz])
                a.append([tempx, tempz+dbx_z])
                a.append([tempx+dbx_x, tempz])
            else:
                a.append([tempx+x, tempz])
                a.append([tempx+x, tempz+dbx_z])
                a.append([tempx+x-dbx_x, tempz])
            flag=1
        #第三种情况：判断是否为 五棱柱
        if (flag==0):
            dm_s1=x*z-dm_s
            dbx_x=math.sqrt(abs(2*dm_s1/math.tan(abs(dalt)*math.pi/180.0)))
            dbx_z=dbx_x*math.tan(abs(dalt)*math.pi/180.0)
            if((dbx_x<=x) and (dbx_z<=z)):
                if(dalt>0):
                    a.append([tempx, tempz])
                    a.append([tempx+x, tempz])
                    a.append([tempx+x, tempz+z-dbx_z])
                    a.append([tempx+x-dbx_x, tempz+z])
                    a.append([tempx, tempz+z])
```

```
                else:
                    a.append([tempx, tempz])
                    a.append([tempx, tempz+z-dbx_z])
                    a.append([tempx+dbx_x, tempz+z])
                    a.append([tempx+x, tempz+z])
                    a.append([tempx+x, tempz])
                flag=3
        #第二种情况：四棱柱
        if (flag==0):
            dbx_z=(dm_s-0.5*x*x*abs(math.tan(abs(dalt)*math.pi/180.0)))/x
            a.append([tempx, tempz])
            a.append([tempx+x, tempz])
            if(dalt<0):
                a.append([tempx+x, tempz+dbx_z+x*abs(math.tan(abs(dalt)*math.pi/180.0))])
                a.append([tempx, tempz+dbx_z])
            else:
                a.append([tempx+x, tempz+dbx_z])
                a.append([tempx, tempz+dbx_z+x*abs(math.tan(abs(dalt)*math.pi/180.0))])
            flag=2
        xx, zz=get_centerpoint(a)
        return [xx, tempy, zz]
def return_zx_sum(x, y, z, zx_x, zx_y, zx_z, m, dalt):
    xyz_zb=[]                #每个油箱的重心
    yxz_zx=[]                #油箱系统的重心
    sum_m=0
    for i in range(0, len(m)):
        sum_m+=m[i]
    if(dalt==0):
        for j in range(0, len(m)):
            xyz_zb.append(return_zx(x[j], y[j], z[j], zx_x[j], zx_y[j], zx_z[j], m[j]))
        x1=y1=z1=0
        for j in range(0, len(xyz_zb)):
            x1+=xyz_zb[j][0]*m[j]
            y1+=xyz_zb[j][1]*m[j]
            z1+=xyz_zb[j][2]*m[j]
        yxz_zx.append(x1/(sum_m+m_fj))
        yxz_zx.append(y1/(sum_m+m_fj))
        yxz_zx.append(z1/(sum_m+m_fj))
    else:
        for j in range(0, len(m)):
            xyz_zb.append(return_zx_dalt(x[j], y[j], z[j], zx_x[j], zx_y[j], zx_z[j], m[j], dalt))
        x1=y1=z1=0
        for j in range(0, len(xyz_zb)):
            x1+=xyz_zb[j][0]*m[j]
```

```
                y1+=xyz_zb[j][1]*m[j]
                z1+=xyz_zb[j][2]*m[j]
            yxz_zx.append(x1/(sum_m+m_fj))
            yxz_zx.append(y1/(sum_m+m_fj))
            yxz_zx.append(z1/(sum_m+m_fj))
    return yxz_zx
#数据的读取
XX1=pd.read_excel("附件 5-问题 4 数据.xlsx", sheet_name="发动机耗油数据")
XX2=pd.read_excel("附件 5-问题 4 数据.xlsx", sheet_name="飞行器俯仰角")
YH=XX1["耗油速度(kg/s)"].values
FY_dalt=XX2["俯仰角(度)"].values
#油箱长宽高参数
yx_x=[1.5, 2.2, 2.4, 1.7, 2.2, 2.4]   #长
yx_y=[0.9, 0.8, 1.1, 1.3, 1.2, 1]     #宽
yx_z=[0.3, 1.1, 0.9, 1.2, 1, 0.5]     #高
#油箱中心坐标
yz_x=[8.91304348, 6.91304348, -1.68695652, 3.11304348, -5.28695652, -2.08695652]
yz_y=[1.20652174, -1.39347826, 1.20652174, 0.60652174, -0.29347826, -1.49347826]
yz_z=[0.61669004, 0.21669004, -0.28330996, -0.18330996, 0.41669004, 0.21669004]
#油箱初始油量
yc=[0.3, 1.5, 2.1, 1.9, 2.6, 0.8]
yc_m=[]              #初始油的质量
for i in range(len(yc)):
    yc_m.append(yc[i]*p_ranliao)
def chanshen_bianliang(YH, mi, ti):          #返回变量值一基于遗传
    t=1.0
    m_t=1e-1          #油量限制
    py_l=5e-3    #排油限制
    ub=[1.1*t, 1.8*t, 1.7*t, 1.5*t, 1.6*t, 1.1*t]
    while(1):
        xi=[0, 0, 0, 0, 0, 0]
        flag=1
        ei=[0, 0, 0, 0, 0, 0]
        ai=[0, 5]
        if(mi[0]<m_t):ai.remove(0)
        elif(ti[0]>=1 and ti[0]<=60):ai.remove(0);ei[0]=1;
        if(mi[5]<m_t):ai.remove(5)
        elif(ti[5]>=1 and ti[5]<=60):ai.remove(5);ei[5]=1;
        aj=[1, 2, 3, 4]
        if(mi[1]<m_t):aj.remove(1)
        elif(ti[1]>=1 and ti[1]<=60):aj.remove(1);ei[1]=1;
        if(mi[2]<m_t):aj.remove(2)
        elif(ti[2]>=1 and ti[2]<=60):aj.remove(2);ei[2]=1;
        if(mi[3]<m_t):aj.remove(3)
```

```
elif(ti[3]>=1 and ti[3]<=60):aj.remove(3);ei[3]=1;
if(mi[4]<m_t):aj.remove(4)
elif(ti[4]>=1 and ti[4]<=60):aj.remove(4);ei[4]=1;
sum_gy=0;sum_zg=0
for i in range(1,5):
    if(ei[i]==1):
        sum_gy+=1
for i in [0,5]:
    if(ei[i]==1):
        sum_zg+=1
if(sum_gy==0):
    if(YH<ub[1] and YH<ub[2]and YH<ub[3]and YH<ub[4]):
        if(YH!=0):shenyu1=random.randint(1,2)
        else:shenyu1=0
    else:shenyu1=2
elif(sum_gy==1):
    if(YH<ub[1] and YH<ub[2]and YH<ub[3]and YH<ub[4]):shenyu1=random.randint(0,1)
    else:shenyu1=1
elif(sum_gy==2):shenyu1=0
if(shenyu1>len(aj)):shenyu1=len(aj)
if(shenyu1!=0):
    for i in range(0,shenyu1):
        a=random.choice(aj)
        aj.remove(a)
        ei[a]=1
if(sum_zg==0):shenyu2=random.randint(0,1)
else:shenyu2=0
if(shenyu2>len(ai)):shenyu2=len(ai)
if(shenyu2!=0 and len(ai)!=0):
    for i in range(0,shenyu2):
        if(mi[0]<=m_t or mi[5]<=m_t):
            if(mi[5]>m_t or mi[0]>m_t):
                if (mi[0]<m_t):
                    ei[5]=1
                elif(mi[5]<m_t):
                    ei[0]=1
            else:
                break
        else:
            a=random.choice(ai)
            ai.remove(a)
            ei[a]=1
flag_1=[]
flag_2=[]
```

```
            flag_3=0#跳出循环标志
            for i in range(0, len(ei)):
                if(i==0 or i==5):
                    if(ei[i]==1):flag_2.append(i)
                else:
                    if(ei[i]==1):flag_1.append(i)
            if (len(flag_1)==0):pass
            elif(len(flag_1)==1):
                if(mi[flag_1[0]]>=ub[flag_1[0]] and YH<mi[flag_1[0]]):
                    xi[flag_1[0]]=random.uniform(YH, YH+py_l)
            elif(len(flag_1)==2):
                if(YH!=0):
                    while(xi[flag_1[0]]<=0 or xi[flag_1[0]]>=ub[flag_1[0]] or xi[flag_1[1]]<=0 or
xi[flag_1[1]]>=ub[flag_1[1]]):
                        if(mi[flag_1[0]]>YH):
                            xi[flag_1[0]]=random.uniform(0, YH+py_l)
                            if(flag_3>=500):
                                ti[flag_1[1]]=0
                        else:
                            xi[flag_1[0]]=random.uniform(0, mi[flag_1[0]])
                            if(flag_3>=500):
                                ti[flag_1[0]]=0
                        xi[flag_1[1]]=random.uniform(YH, YH+py_l)-xi[flag_1[0]]
                        flag_3+=1;
                        if(flag_3>=500):break
            if(flag_3>=500):continue
            for m in flag_2:
                if(mi[m]>ub[m]):
                    xi[m]=random.uniform(0, ub[m])
                else:
                    xi[m]=random.uniform(0, mi[m])
            for i in range(0, len(mi)):
                if(mi[i]-xi[i]>=0):flag=flag&1
                else:flag=flag&0
            if(flag==1):
                break
        return xi
ti=[0, 0, 0, 0, 0, 0]
xi=[0, 0, 0, 0, 0, 0]
FAn_60s=[]          #供油变化量
YH_sum=[]           #油耗和
OBJ=[]
t=1
for k in range(0, int(len(YH)/t)):
```

```
    yh_SUM=0
    obj=100
    for j in range(0, t):
        yh_SUM+=YH[t*k+j]
    print(yh_SUM)
    YH_sum.append(yh_SUM)
    for j in range(0, 1000):
        m_temp=[]
        x_temp=chanshen_bianliang(yh_SUM, yc_m, ti)
        for i in range(0, len(x_temp)):
            m_temp.append(yc_m[i]-x_temp[i])
            if(i==1):
                m_temp[1]+=x_temp[0]
            if(i==4):
                m_temp[4]+=x_temp[5]
        zx=return_zx_sum(yx_x, yx_y, yx_z, yz_x, yz_y, yz_z, m_temp, FY_dalt[k])#求取质心
        obj_temp=math.sqrt((0-zx[0])**2+(0-zx[1])**2+(0-zx[2])**2)
        if(obj_temp<obj):
            obj=obj_temp
            out_zx=[];out_x=[];out_m=[]
            for m in  zx:
                out_zx.append(m)
            for m in x_temp:
                out_x.append(m)
            for m in m_temp:
                out_m.append(m)
        del zx, x_temp, m_temp
    yc_m=[]
    for m in out_m:
        yc_m.append(m)
    print(k)
    for i in range(0, len(out_x)):
        if(out_x[i]!=0):
            ti[i]+=1
        else:
            ti[i]=0
    print(out_zx);print(obj)
    print(out_x);print(out_m);print(ti)
    FAn_60s.append(out_x)
    OBJ.append(obj)

yx1=[];yx2=[];yx3=[];yx4=[];yx5=[];yx6=[]
for z in FAn_60s:
    yx1.append(z[0]);yx2.append(z[1]);yx3.append(z[2])
```

```
    yx4. append(z[3]);yx5. append(z[4]);yx6. append(z[5])
XX3=pd. DataFrame()
XX3["1 号"]=yx1;XX3["2 号"]=yx2;XX3["3 号"]=yx3
XX3["4 号"]=yx4;XX3["5 号"]=yx5;XX3["6 号"]=yx6
XX3["目标"]=OBJ
print("最大偏差：", np. max(np. array(OBJ)))
XX3. to_excel("供油曲线 4_10. xlsx")
```